AFRL-SR-BL-TR-98-

0012

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12/19/97	Final Technical Report 6/1/95 - 9/30/97

4. TITLE AND SUBTITLE

OBJECT DESCRIPTION AND RECOGNITION IN COMPLEX ENVIRONMENTS

5. FUNDING NUMBERS

F49620-95-1-0457

6. AUTHOR(S)

R. Nevatia, G. Medioni and K. Price

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Southern California
Institute for Robotics & Intelligent Systems
Powell Hall 204
Los Angeles, CA 90089-0273

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NM
110 Duncan Ave. Room B115
Bolling AFB, DC 20332-8050

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This final report describes our work on three topics. First is construction of 3-D models by use of smart 3-D pen which enables direct 3D input and manipulations. The other two efforts deal with the problem of aerial image analysis. First of these is about detecting changes in man-made structures using a paradigm of comparing new images to 3-D models constructed from previous images rather than the previous images directly. The second is about detecting and describing building stuctures from multiple views using hierarchical grouping and matching techniques to overcome ambiguity problems.

14. SUBJECT TERMS

Change detection, model construction and 3-dimensional description

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Object Description and Recognition in Complex Environments

Grant No F49620-95-1-0457

Final Technical Report

June 1, 1995 to September 30, 1997

Contractor: University of Southern California

Start Date: 6/1/95

Principal Investigator: Ram Nevatia

(213) 740-6427

Program Manager: Abraham Waksman

(202) 767-5025

R. Nevatia , G. Medioni and K. Price (Editors)

Institute for Robotics and Intelligent Systems

School of Engineering

University of Southern California

Los Angeles, CA 90089-0273

December 1997

Sponsored by Defense Advanced Research Projects Agency and by Air Force Office of Scientific Research Under Grant No F49620-95-1-0457.

Table of Contents

List of Figures

Preface

This report summarizes research for Grant No F49620-93-1-0620 for the 12 months from March 1996 to September 1997. We give a general overview of the work of the last year in the introduction with more detail provided in later chapters. This report and the previous Annual report constitute the Final Technical Report for the Grant.

1 Introduction

Our effort in the last period has focused on three topics. First is construction of 3-D models by use of smart 3-D pen. The other two efforts deal with the problem of aerial image analysis. First of these is about detecting changes in man-made structures. The second is about detecting and describing building structures from multiple views. A brief summary of these efforts is provided here; details may be found in the following chapters.

1.1 3DSketch

Three-dimensional models are fundamental for many applications such as design, analysis, visualization, animation, and multimedia presentation. The difficulty in building 3D models comes from three aspects: the input devices are two dimensional, the output devices are also two-dimensional, and the 3D modeling tasks usually require artistic skills and computer experience. To tackle these problems, we use a 3D input device for direct 3D input and manipulations; for the output, we currently use projective display with many depth cues such as a floor mesh, shading, occlusion, etc. To make the modeling easier for a common user with no artistic or computer skills, we base our system on a conceptual model of freehand sketching. See Chapter 2.

1.2 Change Detection

In this effort, we address the problem of detecting changes in man-made structures by using aerial images. Changes in image intensities do not always correspond to structural changes as pixel intensities can change with changes in viewpoint, illumination and atmospheric conditions. Also, not all changes on the ground are interesting, for example, some changes may be seasonal and not structural. We use a paradigm of comparing new images to 3-D models constructed from previous images rather than the previous images directly. We have developed techniques for image to model registration, model validation and change detection. We are able to detect changes such as missing buildings, buildings whose dimensions have changed or buildings that have been mismodeled (in dimension or position). Results on real data over the Ft. Hood site are presented in Chapter 3.

1.3 Building Detection and Description

Automatic construction of 3-D models of buildings from aerial images is important for a number of tasks. This problem is made difficult due to problems of low level segmentation, lack of direct 3-D information in 2-D images and complexity of images and the scene in urban environments. We describe a system that detects rectilinear buildings using multiple images. The images need not be all taken at the same time. We use hierarchical grouping and matching techniques to overcome ambiguity prob-

lems in matching of multiple image features. Good results can be obtained on buildings with adequate contrast and limited occlusions. Examples from the Ft. Hood site are used to illustrate the performance of the system. See Chapter 4.

2 3DSketch: Modeling by Digitizing with a Smart 3D Pen

Song Han and Gérard Medioni

2.1 System Overview

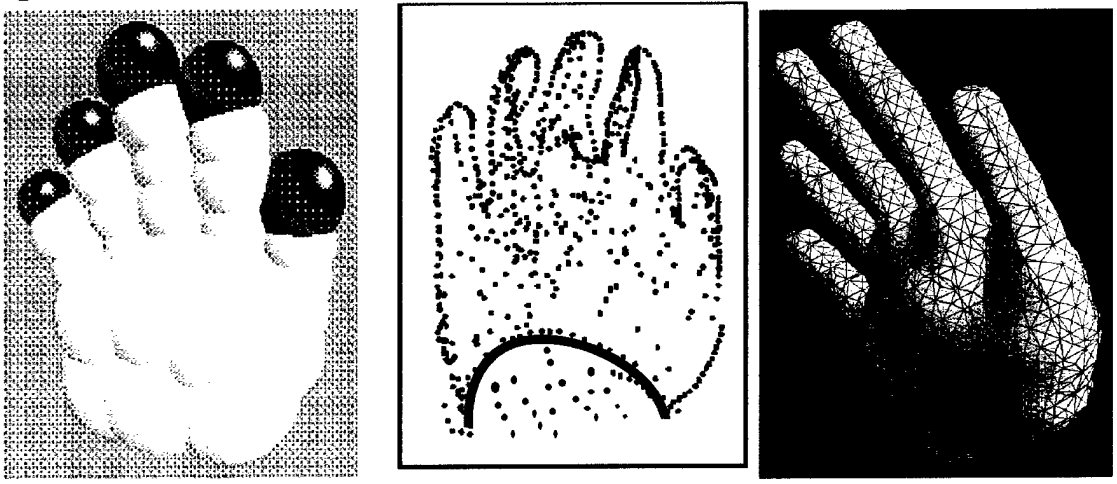
3D models are fundamental for many applications such as design, analysis, visualization, animation, and multimedia presentation. However, building 3D models is not an easy task. The difficulty comes from three aspects: firstly, the input devices (mouse and keyboard) are two dimensional and lack the freedom for a user to specify and manipulate in three dimensions; secondly, the output devices (screens) are also two-dimensional and do not provide enough cues for exact alignment and depth perception; thirdly, the 3D modeling tasks usually require artistic skills and computer experience which only special groups of people may have.

To tackle the above problems, we use a 3D input device for direct 3D input and manipulations; for the output, we currently use projective display with many depth cues such as floor mesh, shading, occlusion, etc. A stereo display device is preferred and will be used later. To make the modeling easier for a common user with no artistic or computer skills, we base our system on a conceptual model of freehand sketching. Our system is novel in that the user only needs to draw unstructured strokes and the computer programs will infer structured representations automatically, and thus the 3D modeling work becomes casual and easy.

Fig. 1. System set-up

Our "3DSketch" system can be used for 3D design, digitizing, and data visualization, but we currently concentrate on 3D modeling by digitizing from a real object. The system set-up is shown in Fig. 1, in which an SGI Indy workstation is used for computation and display, a mouse is used by left-hand for menu/icon clicking and 3D rotation, a 3D pen is held in the right hand, and a 2-button foot pedal is used to pause/continue/stop the data sampling procedure.

The software has three major modules: Prototyper, Refiner, and autoTracer. With the Prototyper module, the user quickly builds a "clay" prototype by sketching a few strokes. With the Refiner module, the user locally improves the prototype surface and aligns edge/corner features by adding more strokes. The autoTracer module automatically finds creases and corners from the stroke data over smooth regions, so that the user can avoid the tedious tracing of such features in the Refiner module. The automatically traced features are then fed into the Refiner module to improve the model. Fig. 2 shows an example.



prototyper auto tracer refiner

Fig. 2. Three modules of the system

From the programmer's view, the clay prototype modeling is implemented as iso-surfacing of a 3D scalar field. The local refinement is a procedure of deformable surface fitting and active edge alignment, using energy minimization in potential fields. The automatic tracing of discontinuity edges and corners are based on directional diffusion and topology analysis of a 3D tensor field.

We implemented the user interface using Xt/Motif and OpenGL. We do not use the 3D pen as a menu/icon selection device, since that will require extra motion of the right hand which does not always rest on the desk. The left hand is always resting on the desk, so hand fatigue is not a problem and the menu/icon selections are allocated to the left hand. Also we want to make use of the Motif utilities as much as possible, instead of writing very low level interactions from scratch using the 3D pen. However,

we have made efforts to reduce the left hand operations by introducing gesture recognition in the system, so that some menu/icon selections are not really necessary.

A "virtual" pen is also displayed on the screen to mimic the actual motion of the 3D pen. The user can look at the pen to check the orientation of the coordinate system, the display mode, or to test lighting and rendering parameters on the pen for fast feedback. If the left button of the foot pedal is pressed, the data digitizing begins. If it is released, the digitizing pauses. Pushing the right button will finish a session of digitizing and back to viewing and modeling mode. If the pen stays static or moves very little, the data recording program will not update the data input, so the user can pause during sketching. A screen shot of the interface is attached in the end of the paper.

2.2 Motivation

We humans can effectively perceive a smooth world and detect discontinuities from noisy sparse scattered samples. There are some very general principles in our human perception processes to account for the regularities in the visual world. If we can convert such perception principles into computer programs, the computer will automatically infer and reconstruct a complete (or nearly complete) model from a compact set of samples. This can remove or significantly reduce the laborious manual overhead in the dense sampling or rigorous structuring in building computer models. Our goal is to free the sketcher from such difficult tasks and allocate them to computers. The tasks allocated to the users are limited to some high level decisions and a few essential operations.

Based on this philosophy, we started to develop the 3DSketch system using a 3D digitizing pen. We call the system "3DSketch" since the strokes are sparse, only giving a "sketchy" description of the object. We demonstrate how to reduce the user's work by making the 3D pen more "intelligent". The user would feel as if he/she had a "smart pen" that can understand his/her intent from the quick sketch and shows a regularized model, with degeneracy automatically detected and marked.

Since current computer modeling tools require precise structured operations, the tools block the user's flow of ideas and interfere with the user's concentration on creativity. Artists and designers still prefer to make freehand sketches on paper with a pen for quick feedback in visual thinking. Some researchers tried to convert such hand sketches to computer models. Lipson and Shpitalni [1] implemented a system to reconstruct 3D models from freehand sketches. The problem with these systems is that the 3D models are inferred from a finished entire 2D sketch, so a lot of intermediate information, such as the orientation of a curve, the sequence between a few curves, the grouping of strokes for a part, and the outer boundary of an object, is already lost. The inference system must perform segmentation, recognition, reconstruction, and representation tasks to interpret the entire finished sketch, and these tasks are often coupled together, making the problem very hard to solve. To avoid such prob-

lems, an interactive system is preferred to perform incremental inference while the sketching is being done, and thus the intermediate information can be utilized. *In our system, intermediate information is used and timely feedbacks are provided.*

2.3 Interactive Sketching Techniques

Most interactive 3D design packages use curve-based sketching methods, such as sweeping, revolving, and extrusion. These methods usually produce quadrilateral meshes. To make a whole object, several patches have to be aligned, with possible gaps and holes filled by more operations. Since the input device is often a 2D mouse, the user switches between XY, YZ, and ZX planes to draw the curves, and the user operations are specified by menus, icons, keyboard shortcuts and direct manipulations. Zeleznik, Herndon and Hughes developed an interactive system SKETCH [6], which does not use any menus or icons, instead the system automatically infers hand gestures from the 2D mouse strokes to determine the solids or surfaces the user intends to input. Adding some hand gestures significantly makes the system easier and faster to use, but totally getting rid of menus/icons may be impractical since there are too many possible gestures in the sketching procedure. In the Quick-sketch system made by Eggli et al. [2], the user sketches over a pressure-sensitive screen with a pen, and the system provides icon highlighting feedback, and user correction is allowed to override the automatic gesture recognition. We use gestures as well as menus/icons in our 3DSketch system, but our sketching device is a 3D pen, and the system has more self-adaptation and spatial reasoning ability.

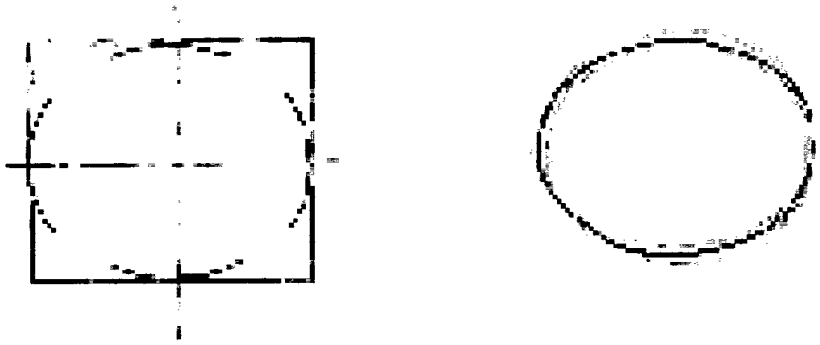
In other related work, 3D devices are also used to replace the 2D mice. Sachs et al. [4] described a "3-draw" system. The user holds a mirror-like plate in the left hand to rotate the whole screen display, and holds a 3D stylus in the right hand to sketch curves in 3D space. Deering [5] implemented a virtual environment called "holosketch". Despite the existence of so many 3D modeling software packages, we find that many producers still prefer to start with clay models because clay is malleable and allows artists the full flexibility to manipulate surfaces in ways that are not easy to use or not available in off-the-shelf modeling softwares. Many methods and steps are involved in translating a clay model into a computer model. Our goal is to make the translation more efficient. Our system is special in that it allows random scratching over a region, so that rigorous curve tracing or structured patch meshing is not required. Such random scratching is much easier and faster to use than the conventional curve-based designing schemes. For the latter, the user cannot pause during sketching a curve, and must follow a restrictive sequence to specify a surface mesh, mostly a rectangular mesh which has artifacts in modeling irregular shapes and corners.

Many modeling activities, such as sketching and sculpting, involve both hands. According to Guiard's psychophysical research [3], the left and right hands often act as elements in a kinematic chain. For right-handed people, the left hand acts as the base link of the chain. The right hand's motions are based on this link, and the right

hand finds its spatial references in the results of the motion of the left hand. Also, the right and left hands are involved in asymmetric temporal-spatial scales of motion: right hand for high frequency fine motion, left hand for low frequency coarse motion. Our 3DSketch system uses this natural division of manual labor by assigning the low-frequency coarse setting of spatial context to the left hand, and the high-frequency fine selection and manipulation operations to the right hand. The left hand provides context by doing menu/icon selection, rotating the global scene, and providing constraint mode. The right hand operates in the frame of reference set up by the left hand, and operates after the left hand's selection of a command and picking of a tool. The right hand performs finer sampling, sketching and manipulation of the surface geometry.

Although the 3D pen can also be used for designing surfaces from imagination or fitting surfaces to real data such as CT/MRI medical data, we are now just concentrating on modeling by digitizing an existing object. Some subtle differences may exist between these applications. For example, a sphere can be designed by clicking at the center then stretching an initial sphere to a desired radius. But in digitizing, the probe can never move inside a solid object to specify a center. For digitizing, one display view is enough, but for modeling and visualization of real data, multiple display views may be necessary for 3D alignment.

To digitize curved, especially organic shapes, current systems require that the user trace very dense and regular meshes on the surface. Such work is tedious and time-consuming (8 hours are reportedly needed for an experienced modeler to manually complete a surface mesh by digitizing a toy cat). Our system allows the user to casually specify the most important features and let the computer to fill in the regular gaps.



(a) boxing-up, dimensions, proportions (b) local refinement

Fig. 3. Steps in sketching an ellipse

Textbooks on sketching teach the students to first box-up the objects to draw, outline dimensions and proportions, then refine the details. The simplest example is to sketch an ellipse. First draw a horizontal axis and a vertical axis, then draw a rectangle box, see Fig. 3(a), and small arc segments are drawn at the top, bottom, left and

right locations to provide tangent constraints. From this incomplete sketch, we can already perceive a good ellipse (our eyes or brains fill the gaps based on perceptual experience). To finish the sketch, more strokes inside these key points and segments are filled in, and some local modifications may also be made, see (b).

These techniques provide guidelines for our system design. In our system, the user first makes a very crude prototype (a blob, a plate, a stick, etc.); then adds more strokes to specify surface details, edges, and corners, and the crude surface will automatically deform to reach the sketch strokes and the surface edges and corners will also move to align with the specified edges and corners. Since it is difficult to trace the pen along ridge edges on objects (the pen will often slip off the object), we provide an automatic inference module which can infer edges and corners from the user's sketching strokes over the nearby smooth regions, and thus the user is free from the edge tracing tasks. The automatic inference module is based on our group's previous research on perceptual grouping [18, 9].

In next Section, the principles and the implementation details are given. The work is summarized in Section 5 and the future directions are discussed in Section 6.

2.4 Three Modules of the System

2.4.1 Prototyper: Boxing-up and Outlining

2.4.1.1 User's View: Clay Modeling

In the prototyping module, the user rapidly builds a 3D model giving a crude shape but with the correct topology of the object. We implemented an efficient volumetric clay modeling scheme based on hand gesture recognition, and the program automatically results in a regular dense mesh from the clay pieces. Thus, the user does not need to manually trace the dense mesh as in the currently commercial packages.

An object is either a simple entity or an assembly of several parts, and a part is approximately a blob, a stick, or a plate, and each part can be globally deformed to produce a more general part. To digitize a human head, a single sphere prototype can be produced by just clicking 4 points on the head, then the sphere can be globally tapered, and then locally deformed in the refinement module. To digitize a hand, we can specify the palm, the thumb, and the four fingers separately then glue them together to give a single triangular mesh for later local refinement. A finger may also be prototyped as a cylinder, or as a few shorter jointed cylinders, or as a few glued blobs. Compared with surface patch method, volumetric clay modeling is more efficient and there are no gaps or holes in the parts or the objects, and it does not require Boolean operations in the composition.

Gesture recognition is used for fast modeling without clicking the menus or icons. The recognition rules are as follows: if more than three strokes are sketched, an ellipsoid is generated; if only two strokes, the closed (or almost closed) stroke is taken as cross-section and classified into a square or an ellipse in 3D space, and the other

open stroke is used as the sweeping axis for the cross-section. If both strokes are closed, a negative part is recognized, i.e., a hole is produced. Of course, the user can always override the recognized properties by manually clicking the icons; for example, the user can sketch a closed curve as the cross-section of a hole, and then sweep along the hole to specify an axis, then changes the default "positive" property to "negative". Fig. 4 shows a prototyping example.

To further reduce the menu/icon clicking, we plan to implement more gestures for delete, undo, redo, etc., but the menu/icon options are still available.

2.4.1.2 Programmer's View: Scalar Fields and Iso-surface Extraction

Clay models are simulated as equipotential surfaces (or iso-surfaces) traced from a generated scalar field in 3D space. For example, if a fire is placed at (0,0,0), the temperature at any position may be defined by $t(x,y,z)=f(x^2+y^2+z^2)$, where $f(d)$ is any decreasing function and we use $f(d)=2(d/R)^3-3(d/R)^2+1$ (for $d<R$), *and* $f(d)=0$ *(for* $d>R$*).* Then, given a specified temperature value T, the iso-surface is defined by $t(x,y,z)=T$, which is a sphere. If one more fire is put at a nearby position, the new temperature field will be the summation of the two fields, and the iso-surface $t_1+t_2=T$ will become a peanut-like shape. But for the user, it works as if two soft clay blobs blended into a peanut-like shape. Different field functions define different shapes of clay parts. We use ellipsoids, blocks, and cylinders. Furthermore, if ice (instead of a fire) is put in the space, a negative temperature field will result in a cavity, a hole, or a tunnel. Also, a clay part only blends with parts in the same group. For example, a finger is allowed to blend with the palm but will not blend with other fingers.

2.4.2 Refiner: Adding Details to the Prototype

2.4.2.1 User's View: A Magnetic Pen-tip

Since the prototype only gives a crude initial model, the user still needs to improve the surface by adding more details. This is similar to the previously mentioned ellipse sketching procedure, where more local modification strokes are added to improve the boxing-up prototype. In our system, the user randomly sketches over a region where corrections are needed, and the surface will adapt to the new strokes to yield a better shape. The user can also trace the crease edges and mark corners on the object for the model to align with. Finally a smooth surface with preserved and aligned creases and corners will be obtained. The user feels as if the 3D pen has a magnetic pen-tip which attracts the surface to desired positions. See Fig. 5 for a demonstration.

2.4.2.2 Programmer's View: Potential Fields and Energy Minimization

Since the clay parts are solid models (balls, sticks, cubes) and lack local details, we now treat the prototype as a surface model for local refinement by adjusting the positions of control points. The triangle mesh is updated to a quadratic triangular Bezier surface. Each data point contributes a negative spherical potential field which decays with distance, and thus three potential fields are generated by surface points,

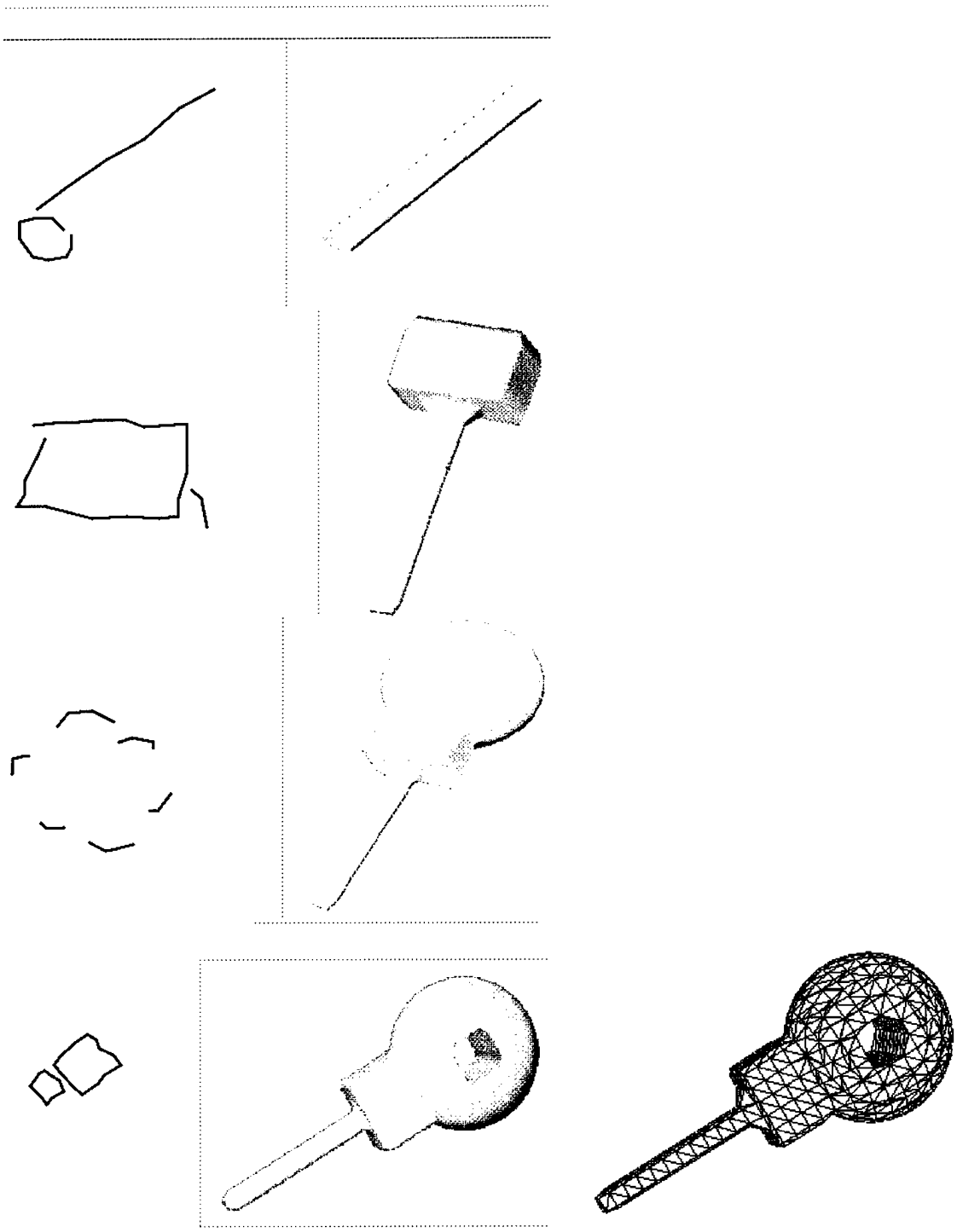


Fig 4. Clay Prototyping by hand gestures

crease points and corner points, respectively. Energy minimization is performed to deform the surface and align its edges and corners. Using potential fields makes surface fitting fast enough for interactive modeling. In Hoppe et al.'s work [10][11], the distances from data points to the triangle mesh is computed many times during the en-

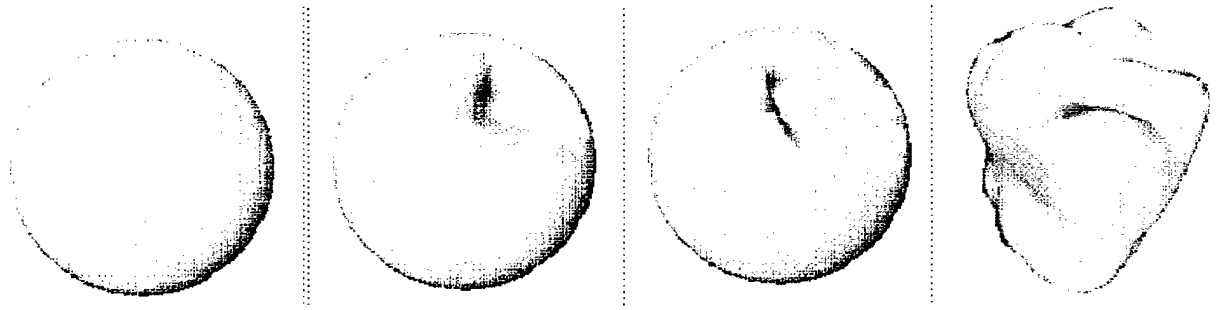


Fig. 5. Refining a blob into a tooth

ergy minimization iterations, and the program runs for hours. By contrast, the potential field is only computed once for each data point, and is incrementally updated with new sketching strokes.

The energy to be minimized is defined as follows:

$$= E_{smooth} + E_{surface} + E_{creases} + E_{corne}$$

At first, the surface is a quadratic Bezier spline surface which is at least C^0 by sharing control points between adjacent triangles, but we still need the smoothness energy to minimize the mesh roughness. The smoothness energy is defined in terms of the first-order derivatives [9]. The surface energy is for surface fitting. Minimizing the crease/corner energy aligns the creases/corners and significantly reduces the total energy, or cooperatively improve the surface fitting precision, and we thus do not have to subdivide the triangles into many tiny ones to obtain a good fitting if misaligned edges and corners are present. After the TriBezier surface is refined, creases and corners are marked and the model is upgraded to C^1 triangular B-spline (TriB) surface with preserved edges/corners *(TriBezier is a special case of TriB)*, one more time of energy minimization is performed to obtain the final surface [9].

We use the Levenberg-Marquardt algorithm with numerically estimated gradients of the energy and take the steepest descent direction in iterations (in physics, the gradient of a potential field is a force field). To maintain interactive speed, the active triangles should be kept as few as possible, so the user must move the pen locally, instead of traversing a long stroke over the object. After a region is refined, the pen can be moved to another local region.

We have recently found that vector potential fields provide better convergence than the scalar potential fields, at the cost of more storage and computation. The reason is that some position may receive conflicting forces from nearby data points. Using vector potential field with radial directions for each data point will cancel out the conflicting forces in the accumulated potential field, making it possible to attract the surface toward correct directions more efficiently.

2.5 AutoTracer: Feature Inference

2.5.1 User's View: Sketching Smooth Regions

In the "Refiner" module, the user quickly and randomly sketches more strokes to improve the model. But, tracing and aligning the creases and corners is still tedious. Valley creases are easier to trace, but the ridge creases are difficult to trace. The pen will slide off the ridges very often. Touching a few points on the ridges is easier but much slower if the ridges are curved. With the "AutoTracer" module, the user casually and quickly sketches over the smooth surface regions, then the computer will automatically infer creases and corners between these smooth regions, and the user doe not need to manually trace the edges or mark the corners. Then the Refiner module is used to perform the energy minimization which deforms the smooth surface and better aligns the creases and corners. Fig. 6 shows the data points produced by sketching over a banana, and the surface with automatically detected and aligned crease edges. The user does not have to trace the edges, making sketching much easier. The inference, however, runs for 3 minutes and thus this module is not done in real-time.

Fig. 6. Strokes over a banana and the result

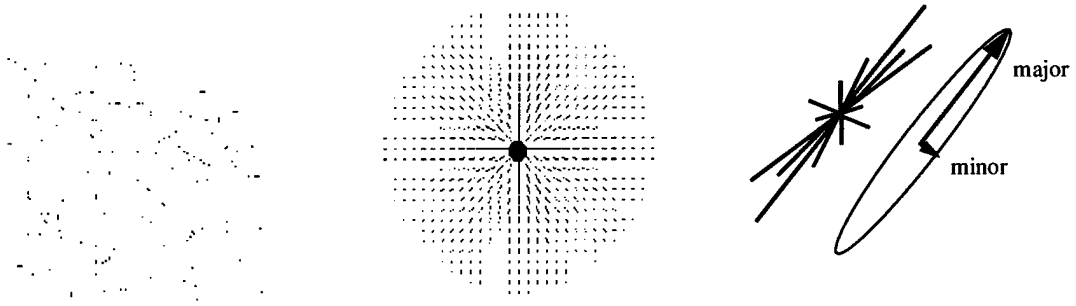
2.5.1.1 Programmer's View: Inkspot Diffusion

From the programmer's view, the problem is to reconstruct smooth surfaces from the scattered points or curve segments, and also to localize creases and corners, if any. If the user sketches slowly, the line segment between two sequential points may provide a good estimation of the tangent of the surface, and a stroke (i.e., a polyline or a fitted B-spline curve) provides a "streamline" over the object (imagine that the user is sketching over a streamlined car hood). However, if the user sketches very quickly, the sampled data points are far apart, and connecting them into a polyline, or fitting them to a spline curve may not yield truthful streamlines of the surface, so more strokes are to be added to obtain enough samples. To make the sketching easy, these strokes do not have to be parallel and the user can change stroke directions while sketching, so the recorded data are just unstructured strokes. To obtain a geometric model of the surface, the programmer's task is to diffuse these streamlines into a stream surface. In addition, the crease edges and corners between these smooth stream surfaces are to be extracted automatically.

To be as general as possible, we only use the sampled unstructured points and diffuse them into a stream surface. This is equivalent to diffusing rain droplets on a

car's hood into a shallow water stream surface and thus obtaining the surface model of the hood. Since we are using a 3D "pen", we'd like to call each data point an inkspot (similar to "streamball" in [13]). To simplify the principles, we first explain the diffusion procedures in 2D, that is, to diffuse inkspots on a sheet of paper into streamlines and detect the corners as well. If the user sketches slowly, the streamlines are readily available by simply connecting sequential inkspots and fitting them with a B-spline curve, and then we can skip the inkspot diffusion step and directly diffuse the streamlines into stream surfaces. Details on the algorithms can be found in [18, 9].

Fig. 7 (a) shows some inkspots with added noise for testing. In (b), an inkspot is shown. The mass density is the largest and the grayscale the darkest (black) at the inkspot position. Then as an inkspot diffuses, its mass density decays with distance and the ink darkness decays also. For a position on the paper, since the ink reaches the position from a specific direction, physical measures such as mass density, velocity, and kinetic energy density are all vectors. For each position, all nearby inkspots can diffuse and reach this position with different mass density along different directions. That means that each position accumulates many mass density vectors, as depicted in (c), and thus each position contains a tensor and the paper becomes a dense 2D tensor field. This is similar to the stress tensor field inside a solid, where each position receives stress forces of different strength and directions from all nearby positions [15,16,17].

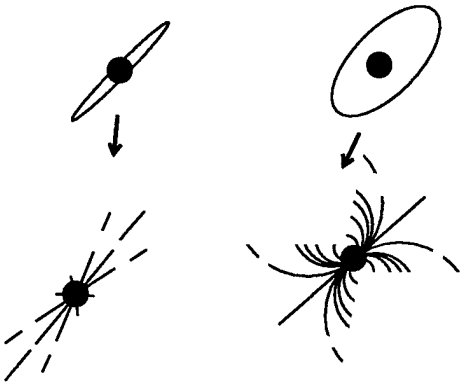


(a) inkspots of a sketch (b) diffusion of inkspot (c) accumulated mass density vectors

Fig. 7. Inkspot diffusion

Mathematically, we compute at each position a 2x2 covariance matrix (or called second-order moments, or scatter matrix) of all accumulated vectors, and the two eigenvectors defining the major and minor axes of an ellipse, as seen in (c). A large thin ellipse indicates an ink stroke passing through this position, and the major eigenvector gives the tangent direction of the stroke; by contrast, a large round ellipse have two salient directions, indicating an intersection of two or more strokes. More specifically, the major eigenvalue yields an absolute measure of ink mass density, and the ratio (eccentricity) tells whether this position is a stroke curve position or an cor-

ner/intersection position, whereas small eigenvalues indicate positions far from the inkspots.



(a) linear diffusion (b) curved diffusion

Fig. 8. Second pass inkspot diffusion

After the above diffusion, we obtain an estimated stroke direction (curve tangent) at each position, and we can further improve the diffusion result by a second pass of diffusion, but this time with an oriented diffusion: the mass density decays with distance and also decays with the offset angle from the tangent. If the tangent direction is very certain (the ellipse is very thin), a linear diffusion pattern in Fig. 8(a) is used. If the tangent direction is not very certain (round ellipse), the stroke at this position is either very curved or at a corner/intersection, so a curved diffusion pattern is used, see Fig. 8(b).

Fig. 9 (a) is the mass density after the second pass diffusion, and (b) is the stroke curves traced by searching for locally darkest positions. The intersection positions (degenerate points) are also marked.

(a) diffused ink (b) traced streamlines with intersections marked

Fig. 9

In 3D space, inkspots or ink streamlines diffuse to yield a 3D tensor field. At each position the tensor is represented by three eigenvectors which depict an ellipsoid. If the ellipsoid is stick-like, the position belongs to a surface since there exists a unique major normal direction; if it is plate-like, this position is along an edge since there are two major normal directions; for a blob, this position is a corner due to the

three conflicting stream surface normal directions. See Fig. 10 for an intuitive illustration.

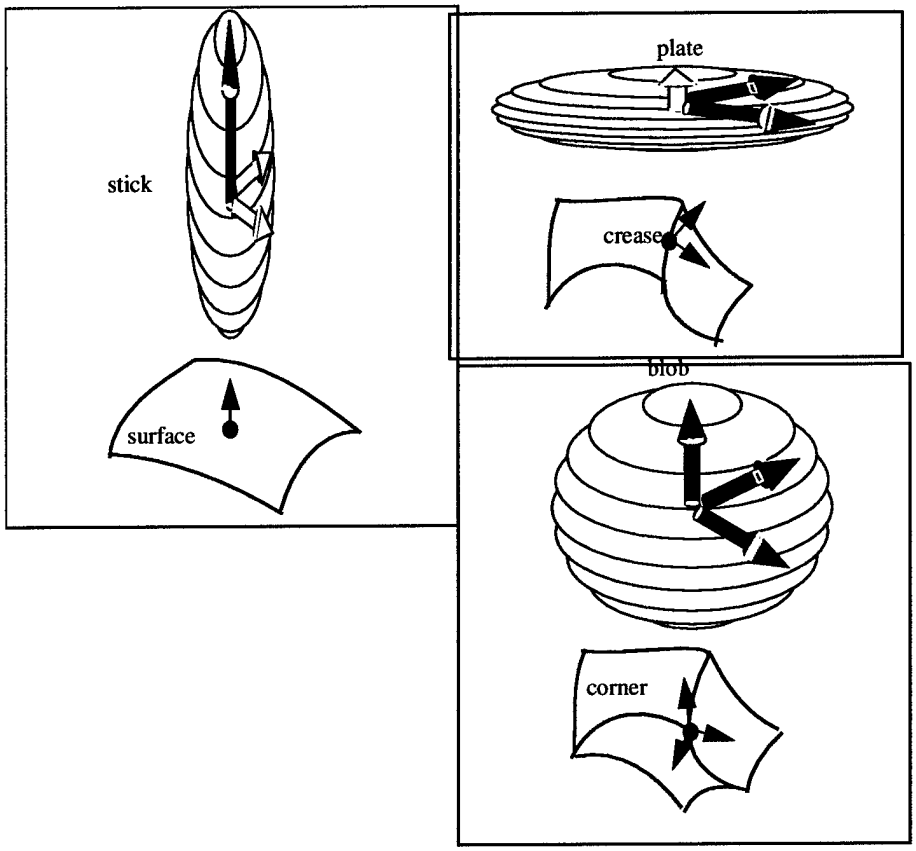


Fig. 10. Eigen analysis of the 3D tensor field

Fig. 11 (a) show 3D strokes sketched over the smooth regions of a wood part. Note that we do not trace any edges or corners. The AutoTracer module infers potential fields for surfaces, edges and corners. A sphere is given by the Prototyper module, and the Refiner module is called to deform the C^0 TriBezier prototype and align the edges and corners. Finally the edges and corners are automatically marked, and the TriBezier surface is upgraded to TriB surface which is C^1 everywhere except along the edges and at the corners, then the Refiner module is called once more for the TriB surface [9].

2.6 Summary

We have described a two-handed 3D "sketching" system using a 3D stylus, and we focus on modeling by digitizing (tracing) an existing object. First, the user simply sketches a few strokes over the object to obtain a 3D prototype; then when the user randomly sketches more strokes over the object and specifies creases and corners, the 3D model will deform to follow the details and the edges and corners will align with the sampled creases and corners. The system can also automatically perform spatial reasoning from unstructured fragmented sketches to infer smooth surfaces and ex-

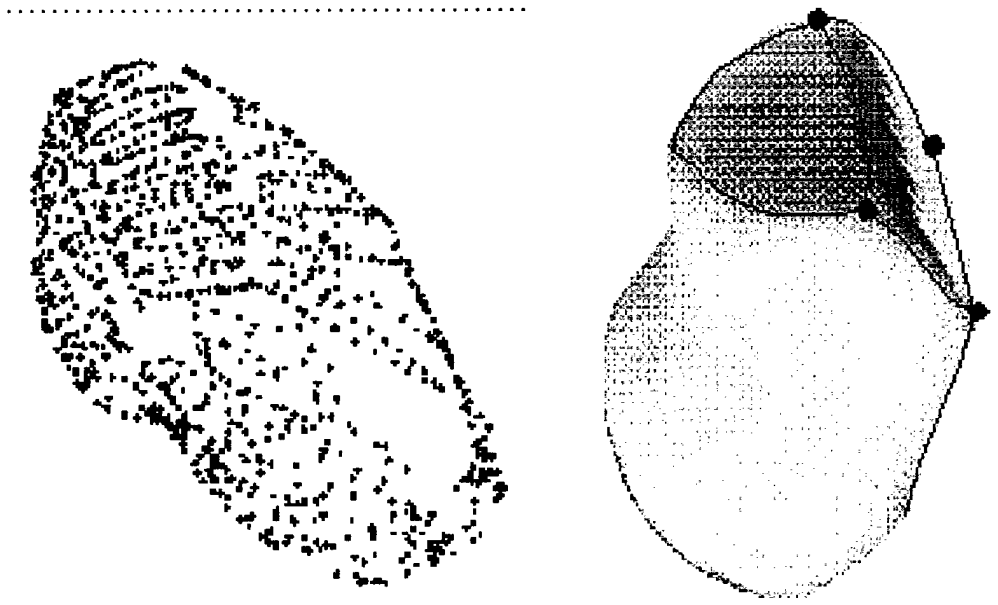


Fig. 11. Autotracer on a piece of wood

tract creases and corners, so that the tedious manual sampling of creases and corners can be avoided.

Technically, the prototyping module uses equipotential surfaces of a scalar field to produce clay-like models; for the local refinement module, scalar or vector potential fields are used to generate attractive forces to deform the shape; for the automatic inference module, tensor fields are used to find surfaces and features (creases and corners).

2.7 Future Work

The internal surface representation is triangular splines, whose good properties in arbitrary triangulation and local subdivision makes it more flexible to model general surfaces than the rectangular splines. Spline models involve much fewer control points than the polygonal models in describing smooth surfaces, and thus are advantageous for iterative minimization and interactive editing. We have used TriBezier, TriB, TriNURBS for open and spherical surface modeling [9]. For arbitrary topology surfaces, we now use triangular Bezier surfaces, but will soon upgrade to triangular B-splines [19]. After the model is refined, some flat regions may have redundant triangles, and a mesh decimation is necessary to simplify the model. The decimation can also yield a multiresolution representation, which is useful for viewing and animation.

For the user interaction, we now use a 2D mouse for menu selection and 3D rotation. We have recently ordered a 6-degree-of-freedom space ball, which will provide more ease of use for the left hand operations. Some company has attached a laser

beam and sensor at the pen tip, so that the object will not be damaged by the touching of the sharp pen tip during sketching. More hand gesture recognition techniques will be applied to further reduce the time necessary for looking at the screens for menus and icons. With a see-through head mounted display, the user could directly see the 3D model imposed on the real object, and find the regions where more refining strokes are needed, and a translucent display which does not block the user's view may be helpful.

2.8 References

[1] H. Lipson and M. Shpitalni, Optimization-based Reconstruction of a 3D Object from a Single Freehand Line Drawing, CAD, vol. 28, no. 8, Aug. 1996

[2] L. Eggli, et al., Inferring 3D Models from Freehand Sketches and Constraints, CAD, vol. 29, issue 2, Feb. 1997, pp. 101-112,

[3] Y. Guiard, Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model, The Journal of Motor Behavior, 19(4), 486-517, 1987

[4] E. Sachs, A. Roberts, and D. Stoops. 3-Draw: A tool for designing 3D shapes. IEEE CG&A, pp. 18-25, Nov. 1991

[5] M. Deering, The Holosketch VR Sketching System, Comm. ACM, vol. 39, no. 5, May'96, pp. 54-61

[6] R. Zeleznik, K. Herndon, and J. Hughes, SKETCH: An Interface for Sketching 3D Scenes, SIGGRAPH'96, pp. 163-170, Aug. 1996

[7] J. Bloomenthal, An Implict Surface Polygonizer, Graphics Gems IV, San Diego: Academic Press, Inc., 1994

[8] W. Lorensen and H. Cline, Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm, Siggraph, Jul. 1987

[9] S. Han and G. Medioni, Triangular NURBS Surface Modeling of Scattered Data, IEEE Vis'96, 295-302, Oct. 1996, San Francisco

[10] M. Eck and H. Hoppe, Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Types, SIGGRAPH'96, pp. 325-334, Aug. 1996

[11] V. Krishnamurthy and M. Levoy, Fitting Smooth Surfaces to Dense Polygon Meshes, SIGGRAPH'96, pp. 313-324, Aug. 1996

[12] L. Forssel, Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution, IEEE Vis'94, pp. 240-247, Washington D.C., Oct. 1994

[13] M. Brill, H. Hagen, H.-C. Rodrian, W. Djatschin, and S. Klimenko, Streamball Techniques for Flow Visualization, IEEE Vis'94, pp. 225-231, Oct. 1994

[14] A. Parkin, Some Problem of Singular Points and Boundary-conditions in the Sketching of Flow Nets, Geotechnique, vol. 44, no. 3, Sep. 1994, pp. 513-518

[15] T. Delmarcelle and L. Hesselink, Visualizing Second Order Tensor Fields with Hyperstreamlines, IEEE CG&A, July 1993, Vol. 13, No 4, pp. 25-33

[16] T. Delmarcelle and L. Hesselink, The Topology of Symmetric Second-order Tensor Fields, IEEE Vis'94, October 17-21, 1994, Washington, D.C., pp. 140-147

[17] L. Hesselink, Y. Levy, and Y. Lavin, The Topology of Symmetric, Second-Order 3D Tensor Fields, IEEE Trans. Vis&CG, pp. 1-11, Vol. 3, No. 1, Jan.-Mar. 1997

[18] G. Guy and G. Medioni, Inferring Global Perceptual Contours from Local Features, Int'l Journal of Computer Vision, vol. 20, no. 1/2, pp. 113-133, 1996

[19] C. Grimm and J. Hughes, Modeling Surfaces of Arbitrary Topology using Manifolds, SIGGRAPH'95, 359-368, 1995

3 Detecting Changes in Aerial Views of Man-Made Structures

Andres Huertas and Ramakant Nevatia*

3.1 Introduction

One of the key applications for aerial and space image analysis is for detecting significant changes on the ground. This could be for various purposes such as urban planning, agricultural analysis, environmental monitoring and military intelligence. We focus on changes in man-made structures, particularly buildings, rather than changes in the vegetation. Note that while changes in the desired structures should be reflected in some changes in the image, not all changes in the images may be caused by 3-D structural changes. Image intensities can vary due to a number of factors such as changes in illumination, viewpoint, atmospheric conditions. Further, seasonal variations may cause changes in vegetation and ground cover; while these represent "real" changes on ground, they may not correspond to structural changes.

There is little previous work in structural change detection. Early work in change detection was based on determining pixel intensity changes [Lillestrand,1972], We believe that the solution to finding structural changes lies in not comparing images taken at different times directly but rather in comparing new images (or descriptions derived from them) to an abstract model derived from previous observations; such models have come to be called *site models* [Gerson & Wood, 1996]. This approach also allows an updating of the site models which may be one of the prime goals of the change detection analysis.

A new image can not be directly corresponded to an abstract model. Instead, we must compute descriptions from the image that can be corresponded with the model or descriptions that can be derived from the model. This is a common problem in object detection in computer vision and various techniques such as *alignment* have been developed to solve it [Huttenlocher & Ullman, 1990]. The change detection problem is simpler to the extent that some parameters of the *pose* of the objects may be known *a priori*. However, the objects themselves may have changed and not fit a prior model exactly. Also, aerial images typically contain a large number of man-made and natural objects, not all of which may have been modeled (for example, we do not assume models for trees in a scene). The objects of interests may be partially (or totally) occluded by other objects and shadows cast by them may cause confusion. The images also contain significant amount of texture, thus leading to a large number of features at the lower-levels (such as edges) that prohibit use of combinatorial techniques to search for desired objects. Finally, we need to verify that the suspected changes actu-

ally correspond to some 3-D structures and to derive a description of the changes where possible.

The system described in this paper is designed for detecting changes in 3-D building structures. We further assume that the buildings are rectilinear and have either flat or simple gable roofs. Composite shapes (such as "L" or "T") are allowed. Each part is represented by its 3-D wire-frame (consisting of any number of vertices and edges). We assume that the camera geometry and approximate viewpoint from which images are taken are known. Specifically, we assume that the errors are such that a *projected* model can be corresponded with the image by *translation* only; this is a reasonable assumption for many imaging situations; errors in other parameters would only affect the registration stage of our system as described below.

The change detection process consists of the following five major steps:

- *Site Model to Image Registration*: This step is to register the new image(s) to the stored site model(s); our system uses line feature matching.
- *Site Model Validation*: This step verifies whether the objects in the site model are present in the new image by comparing predicted features with observed features. A confidence value is computed for each object in the model; low confidence values are likely to represent possible changes to the objects.
- *Structural Change Detection:* In this step, we analyze in more detail possible change in the site indicated in the previous step, and determine if the missing correspondences can be explained by the imaging and viewing conditions or whether evidence exists for actual changes. Our system is able to detect missing (or mis-placed) buildings, buildings with dimensional changes, and new buildings under certain conditions.
- *Site Model Updating*: In this step, 3-D models of changes are constructed where possible. These can be reported to a human analyst and reflected in an updated site model (which can then be used to process new images at the next cycle).

The following sections describe the processing at each step and illustrate with an example. More results and evaluations are described in section [3.6]. Our system has been tested primarily on data available for the Ft. Hood, Texas, site. The site models for our tests were constructed by using tools provided in the Radius Common Development Environment (RCDE) [Strat *et al.* 1992; Fua, 1996]. The kinds of changes we are looking for occur over relatively long periods of time; unfortunately, we were not able to acquire data reflecting such changes for this site. Instead, we have modified the site models which should have the same effect as our system only compares images with site models rather than previous images. This method also provides a check on the accuracy and validity of previous site models. Our system has been ported to an industrial laboratory for possible use in current applications.

3.2 Site Model to Image Registration

The first step is to register a site model to an image. In our task, it is reasonable to assume that the imaging parameters are known to some accuracy and that the errors are such that if a site model is *projected* by the known parameters, its features will correspond with those from the image except for translational errors (which may be quite large, in the order of tens of pixels). The registration problem is then that of determining this translation, which we model as being uniform across the image. We need to decide what features of the models and image should be matched to determine the translation. The models are abstract, 3-D wire frame structures, the image is a 2-D array of intensity values. We have chosen to match lines extracted from the image with the lines projected from the site models by the known (approximately) camera geometry. The line matching technique is adapted from an earlier method [Medioni et al., 1990] and has been described in [Huertas, et. al, 1995].

Figure 3.1 shows the line segments extracted from portion of an image of the site. The model registered with the image is shown in Figure 3.2 . We find this process to be quite robust, whether applied to small windows containing just a few buildings and to very large windows containing many tens of buildings (and other structures which may not be in the site model).

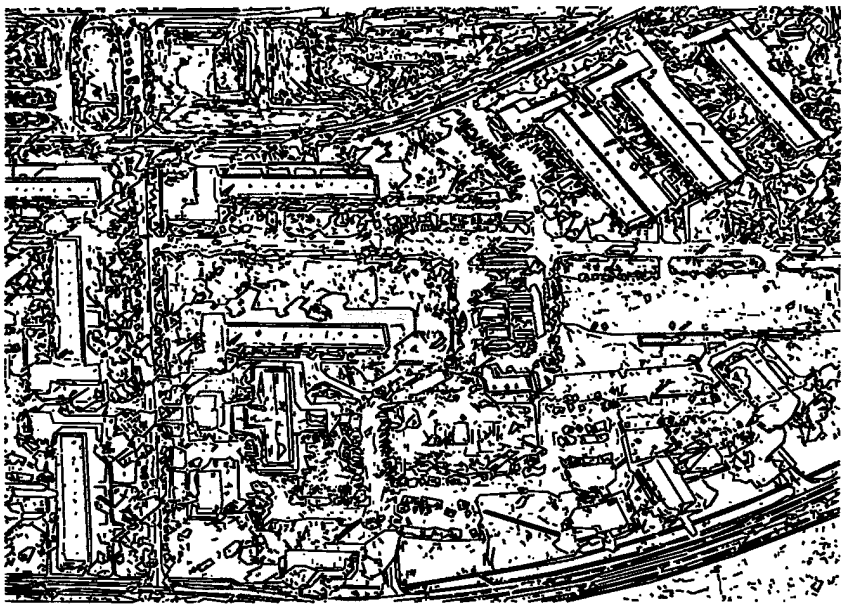


Figure 3.1 Line segments extracted from a portion of an image from Fort Hood, Texas

3.3 Site Model Validation

The purpose of model validation is to verify whether the objects in the site model are present in the new image in the same form or whether they should be examined in more detail for evidence of significant changes. The previous step (registration) pro-

vides a correspondence between model and image segments. For model validation, we combine evidence from a variety of object features such as lines and junctions. We also look for 3-D evidence by evaluating expected shadows cast on the ground; in multiple images, this could come from feature correspondence information. Note that not all the features of the model may be visible in the image, some will be missing due to self and mutual occlusion. These occlusions can, however, be predicted from the viewing geometry and accounted for. There will also be missing evidence due to difficulties of feature extraction in images: low contrast edges may not be detected and line segments fragmented due to surface and ground texture.

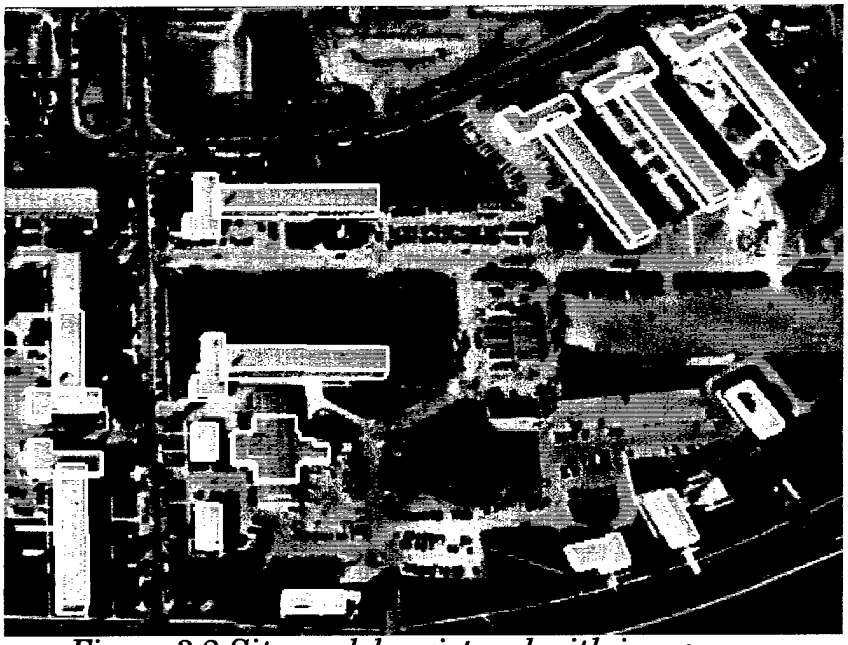


Figure 3.2 Site model registered with image

Before confidence values are calculated (as explained in section [3.3.4]), the system deals with a number of problems and ambiguities inherent to any matching process. We discuss these next.

3.3.1 Missing Features

To validate a model accurately we need to study the source of missing model-to-image correspondences. Some missing image features will be due to viewing conditions such as self-occlusion, occlusion by other objects, self shadows and shadows cast by nearby objects. These, however, can be predicted and explained from the site model itself. Missing correspondences may be due to over- or under-modeling of objects (Figures 3.3 and 3.4) and are more difficult to predict from the model. The confidence associated with over- or under-modeled objects may thus be underestimated or difficult to calculate. Over-modeling is due to the use of modeling primitives that introduce elements that do not correspond to actual physical elements or boundaries. Figure 3.3 shows a building that has been modeled by two rectangle parallelepipeds. The thick

lines represent portions of the elements on the building model that do not correspond to physical boundaries. These can not be matched and the missing correspondences result in lower confidence. Figure 3.4 shows two buildings that are likely to be under-modeled (i.e. modeled by simpler shapes) due to their complexity. These require additional search strategies designed to look for additional and possibly fragmented evidence, such as a large number of vertical or horizontal edge elements. Our system is not currently capable of determining these conditions, and thus the confidence values may be underestimated. Some of these conditions may require annotations in the site model to help the system process these appropriately.

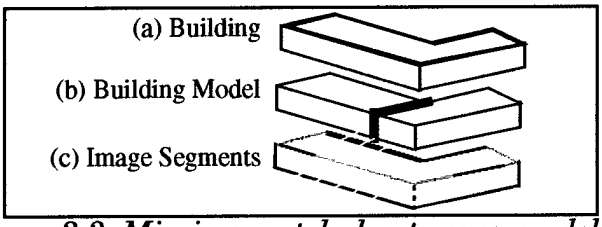


Figure 3.3 Missing match due to over-modeling.

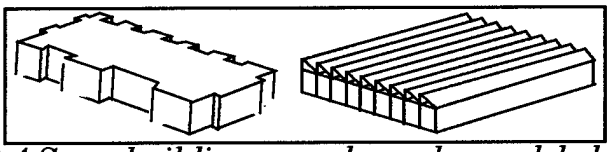


Figure 3.4 Some buildings may be under-modeled

3.3.2 Ambiguities in Matching

The system currently deals with two ambiguities inherent to the matching process: multiple or missing matches, and coincidental alignments due to viewpoint, illumination direction, or due to adjacent structures.

3.3.2.1 Multiple Matches

The model-to-image matcher corresponds each model element with one or more image elements (Figure 3.5) possibly involving more than one object. Allowing multiple matches is necessary to deal with expected fragmentation in the image elements. Fragmentation is due to inadequacies in the feature extraction process and due to actual image content, such as occluding trees, road boundaries and shadows. If a model segment matches multiple colinear image segments, all the image segments are considered to represent image support. If a model segments matches multiple parallel image segments, the overlap among these is considered to represent image support.

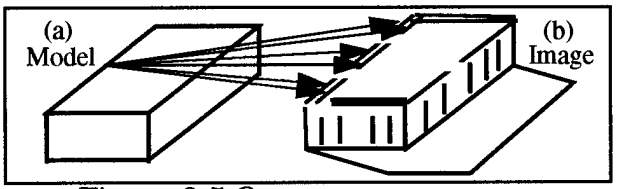


Figure 3.5 One-to-many correspondences.

3.3.3 Coincidental Alignments.

Some multiple matches are due to coincidental alignments of buildings with other structures (Figure 3.6). Some of these include roads, walkways, lawns, shadows and other adjacent objects. Nearby objects and shadows sometimes result in image features that have a larger extent than that predicted by the model features. These are explained by examining nearby shadows with knowledge of the direction of illumination, and by examining adjacent structures. Coincidental alignments due to nearby and adjacent structures are determined by looking for adjacent structures that help explain alignment, or a possible change in horizontal dimensions. An example of some ambiguities and alignments is shown in Figure 3.7 .

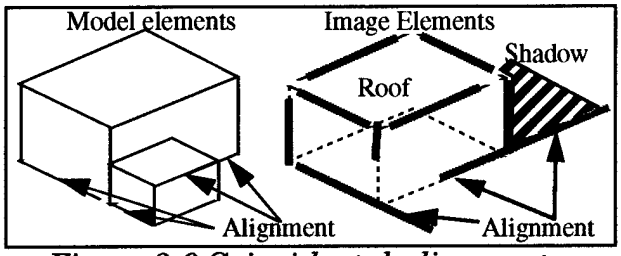


Figure 3.6 Coincidental alignments.

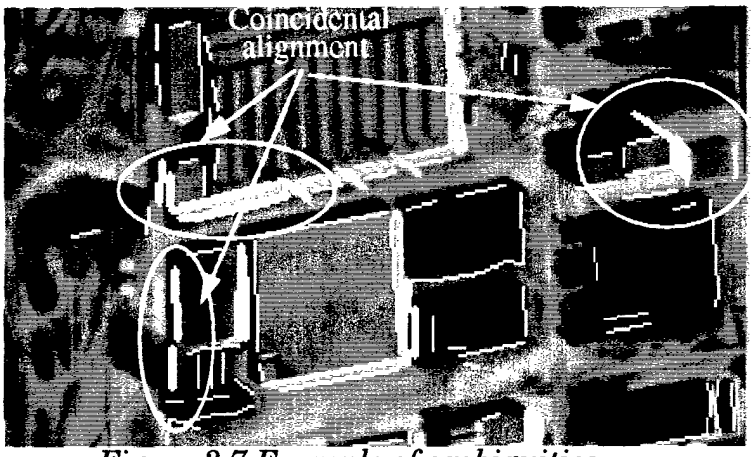


Figure 3.7 Example of ambiguities

3.3.4 Validation Confidence

We evaluate five kinds of evidence: edge visibility, edge presence, edge coverage, junction presence and shadow presence (these terms are explained below). Each evidence provides a score between 0 and 1 and a combined confidence score is computed by a linear weighted combination of them. We have chosen this method of combination for its simplicity. It works well in our tests but there may well be more optimal ways of combining such evidence. Here we have chosen a simple linear combination formulation with empirically assigned weights applied to the evidence terms. The weights however reflect the relative importance of the evidence but remain to be optimized over a larger sample of experiments. The confidence values derived take into account

only visible elements from the particular viewpoint of the image after accounting for self and mutual occlusion. (Figures 3.8 and 3.9):

Let x be a model object defined by a set of vertices and a set of edges. For each object, x, we calculate a confidence value $C(x)$ as a contribution of the following terms:

Edge Visibility: $V(x)$ is given by the fraction of the model edges that are visible from the current viewpoint. Fewer visible edges result in lower confidence.

Edge Presence: $P(x)$ is defined as the fraction of the visible model edges that are matched to some image edges. In the schematic example shown in Figure 3.8 (a), all nine visible edges (dashed lines) have correspondences in the image (solid lines), giving a P value of 1.0. An object that is only 50% visible but that has the visible 50% corresponded to image edges has a P value of 1.0 also. P is calculated separately for **roof** elements, **vertical** wall elements and **base** wall elements which are given different weights (roof evidence is considered the most reliable, the wall base the least reliable).

Edge Coverage: $E(x)$ is defined as fraction of the *lengths* of the visible model edges that is actually covered by some image segments. Figure 3.8 (a) shows an object where all the model edges (dashed) have some, but small coverage; this object has good presence but poor coverage. Figure 3.8 (b) shows the opposite; a few model edges have good image edge support so the coverage is good but presence is not. $E(x)$ is also calculated separately for roof and wall elements. $E(x)$ is penalized by fragmented support.

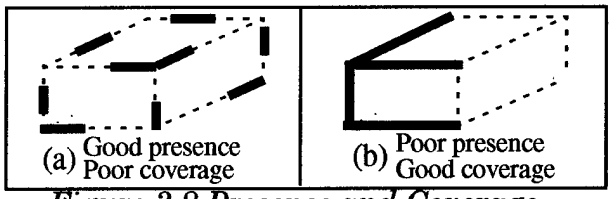


Figure 3.8 Presence and Coverage.

Junction Presence: $J(x)$ is defined as the ratio of the number of image L-junctions at locations predicted by the model (Figure 3.9) to the number of visible model vertices. Image junctions are extracted from the image from the line segments used for matching.

Shadow Presence: $S(x)$, is defined as the ratio of the number of shadow boundaries and junctions extracted from the image matched to predicted shadows, over the number of visible predicted shadow elements (boundaries and junctions) derived from the model (Figure 3.9). See [Lin *et al.*, 1995] for a description of our method to extract shadow boundaries and junctions from images.

High confidence $C(x)$ values indicate good image support while low values denote low support. Low values may signify change as lack of image support may be due to missing buildings, or buildings that have undergone significant change with respect to their current model. However, model buildings that have strong image support,

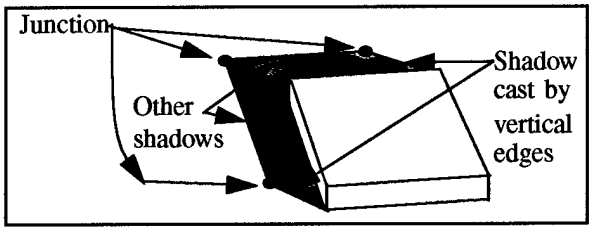


Figure 3.9 Shadows cast by a rectangular building.

may have changed also as additions to structures, such as a new wings, may not affect significantly the appearance of the previously modeled portions.

Figure 3.10 shows the results of the validation step applied to the image shown in Figure 3.2 earlier. The labels indicate the validation confidence level as a function of C values: high (**H**) for $C >= 0.5$; medium (**M**) for $0.4 <= C < 0.5$ and low (**L**) for $C < 0.4$. Note that a building indicated as having low (**L**) confidence has actually drastically changed (or was grossly mis-modeled) whereas the ones marked high (**H**) are in fact, unchanged. Medium level (**M**) typically denote buildings with moderate or "acceptable" image support. These assignments are arbitrary however, and would have to be set as a function of the task at hand. Some applications may require detailed explanations of possible change that require higher discrimination. Here we show only three for simplicity.

3.4 Structural Change Detection

The validation step makes available information that is used to start analyses to determine structural changes. Two cues are used to investigate structural changes:

Validation Confidence values: These values reflect image support for a model object. Although low support may be due to poor image quality, lack of contrast, occlusion, or viewpoint, they can signify missing structures, substantially altered structures or incorrect modeling. Medium level support denote "acceptable" indication of presence with reduced support due to poor image quality, lack of contrast and other image dependent characteristics. High values clearly denote strong presence and image support, at least for the modeled portions.

Extra Image Elements: Model elements that correspond to image elements having greater extent than that of the model element provide preliminary indication of possible changes in dimensions. This situations occur regardless of confidence levels assigned.

The above cues are used to further analyze whether one of the following three classes of structural change has occurred. The three classes are: missing (or misplaced) buildings, dimensional changes, and new buildings. The methods to infer these changes are described next.

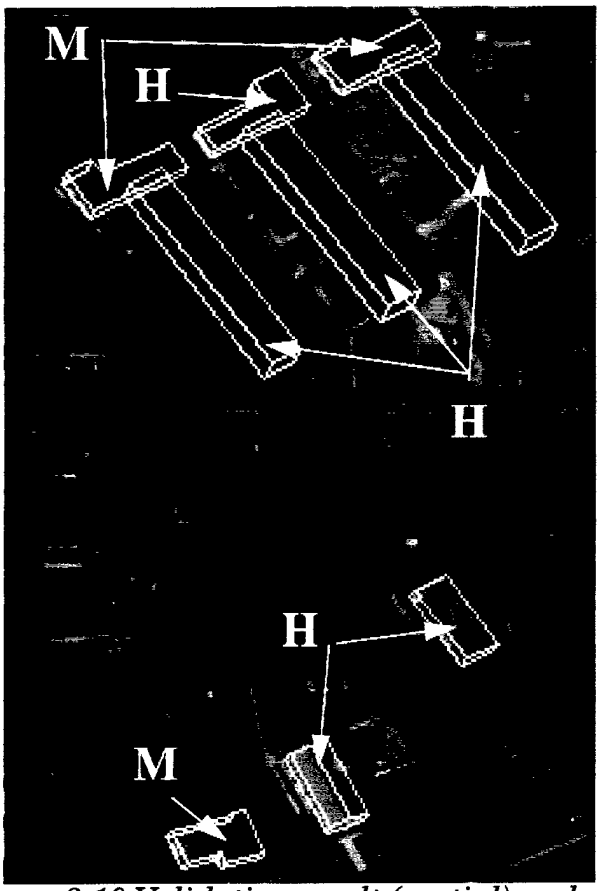


Figure 3.10 Validation result (partial) and confidence levels.

3.4.1 Missing Buildings

Model buildings having very low confidence values denote poor image support. The possible causes for this condition are that either the model is incorrect, the structure is heavily occluded or that the building has been removed or destroyed (assuming that images are of sufficient quality), or that its position is grossly incorrect. A low confidence is sufficient to report a missing building, if additional images were available, they could be examined for confirmation. Figure 3.11 shows the result in a small window containing two modeled buildings. The model building on the right was added to the model by hand to test this condition. It is reported as having low confidence correctly as evidence for its presence can not be found in the image.

3.4.2 Validated Buildings

Model buildings having moderate (**M**) to high (**H**) levels of support are considered validated. That is, their presence in the image is verified. The buildings labeled (M) typically require verification in another image to increase, if possible, their confidence level. An example is shown in Figure 3.12 . The L-shaped building in the small window corresponds to the L-shapes building, labeled M on the top left of the image

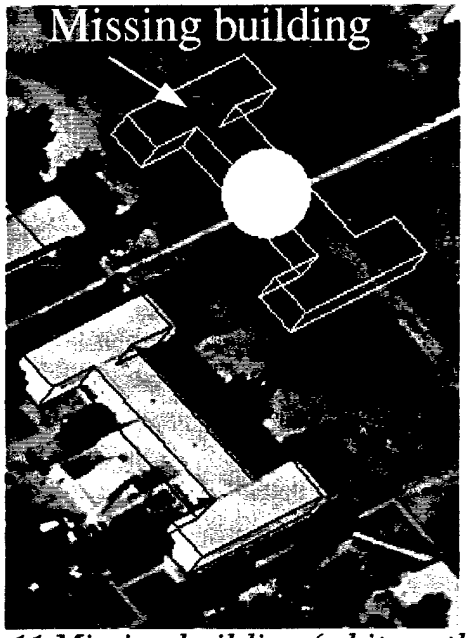


Figure 3.11 Missing building (white outlines)

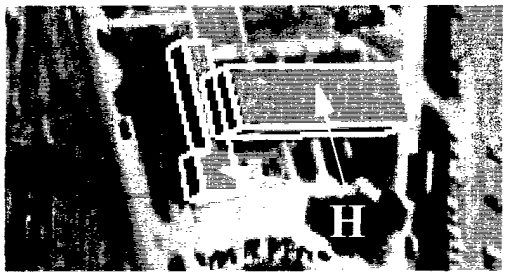


Figure 3.12 Validation from another viewpoint.

shown in Figure 3.10 . From this viewpoint the confidence value of this building increased to 0.5661, thus becoming validated with high confidence.

3.4.3 Dimensional Changes to Modeled Structures

In some cases, regardless of confidence, the system is able to cue possible structural changes based on the fact that model edges are matched to *longer* image edges. Those cases where these conditions arise due to accidental alignments have been handled earlier, as explained in section 3.3.2. The remaining cases therefore represent cues for possible change. We apply a building finding tool at this stage to confirm a change (and to derive a description of the change) as shown later in section 3.5, however, this relies on the building finder's ability to find new buildings. A less stringent criterion may be to only search for some additional evidence such as the extra edges casting a shadow and/or having other evidence of being above ground (from multiple images). We have not implemented such partial analysis though components of it are available in the building finding tool.

Figures 3.13 (a) and (b) show two examples of evidence that support cueing dimensional changes. The matching and fine registration step correctly registers the modified models to the structures in the image. The thick white gray lines are the extended image segments that matched the corresponding model edges thus triggering the cue.

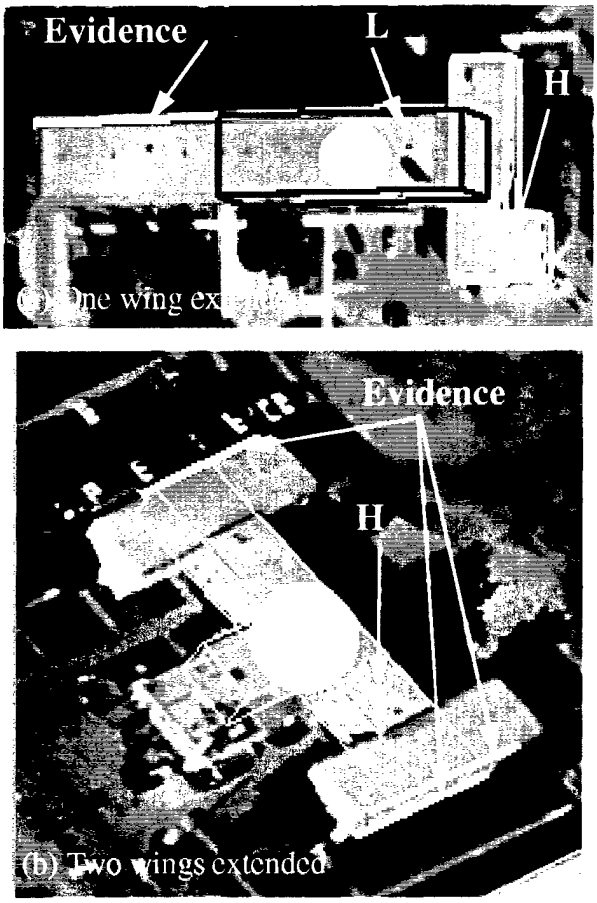


Figure 3.13 Actual change in dimensions.

3.4.4 New Buildings

One important type of site change is the introduction of new structures. For such changes, the previous model is less useful but can still be relied on to provide some context. Such context can consist of areas of interest and characteristics of existing buildings (to check if buildings similar to existing ones have been constructed). Our system only uses the site model to mask out the modeled areas and a building finding tool is applied to all other areas, or all other areas containing large number of unmatched image features such as corners. The camera models and terrain models associated with the site are used to derive viewpoint and illumination parameters automatically. We have experimented with focus of attention mechanisms to select the areas where automated detection should be applied.

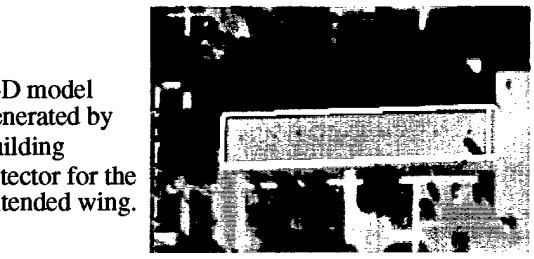
3.5 Model Updating

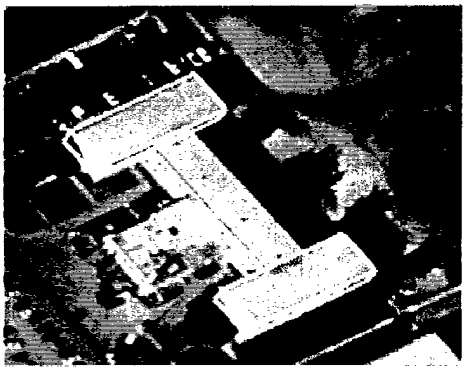
The next task is to make a model for the detected structural changes and to incorporate them in the site model. We describe two situations: where changes are made to existing structures and where new structures are detected.

3.5.1 Modeled Buildings

Changes in the dimensions of modeled structures that have been cued by the previous step need to be analyzed further, possibly using more than one view of the scene. We use a monocular building detection system [Lin *et al.*, 1995] to return the highest rated building hypothesis that can be formed in the location of the cued change. These are shown in Figure 3.14 for the two examples shown earlier in Figure 3.13 . The 3-D models are derived automatically.

(a) 3-D model generated by building detector for the extended wing.





(b) 3-D models suggested for changed wings.

Figure 3.14 Suggested updating of cued changes

3.5.2 New Buildings

Detection of new buildings is a more difficult task as the site model is less helpful. Our method consists of applying one of our building finding tools (see for example [Lin et al, 1995]) to areas of interest and reporting the results that are of sufficiently high confidence. Typically the system would be instructed to locate new buildings in designated areas that are of interest, such as functional areas. The buildings detected automatically become candidates to be added to the site model.

3.6 Validation and Change Cueing Results

We have tested our system on several images of a modelboard and Fort Hood sites. In this section we show part of a representative example only, due to lack of space. Fort Hood images are typically 7775x7720 pixels in size. The 3-D site model contains 79 objects representing building structures. Processing time is about 15 seconds per structure on a Sun sparc-10 workstation, running under the RCDE. The results are summarized in table [1]. It shows the number of building objects visible in the image and the distribution of validation confidence values (**H, M** and **L**). The label codes are also shown in Figure 3.15 . The confidence values are dependent on the image content and may not necessarily reflect structural changes but generally there is a high correlation between confidence level and the number of buildings changed, not changed, or missing. All matching ambiguities, with one exception (not shown), are correctly handled. This case involves an alignment with a ground feature not present in the model, a situation, not currently handled by the system. The building involved is the only one, of the 54 non-changed buildings, cued incorrectly to have changed. This situation is however likely to be corrected by confirmation of the change using another view of the building. Fourteen buildings that are actually present in the image had changes. Thirteen of these are found to be changed correctly. Representative changes are shown in the Figure 3.15 with a circle on top and thick white lines cueing evidence of dimensional changes. The remaining changed building is an L-shaped building (not shown). The building has two wings, both of which have changed. Only the left wing is detected to have changed. This situation is likely to be corrected by reconciling the output from more than one view.

Table 1: Summary of Results

Image (fhov927)	Visible Buildings	Validation Confidence			Non-changed Buildings			Changed Buildings			Missing Buildings		
		High (H)	Medium (M)	Low (L)	Number of buildings	Reported non-changed	Reported changed	Number of buildings	Reported changed	Reported	Number of buildings	Reported missing	Validated
No.	79	54	13	12	54	53	1	14	13	1	11	11	0
%	100	68.3	16.4	15.1	100	98.1	1.9	100	92.8	7.3	100	100	0.0

Buildings that changed considerably or are missing have a low validation confidence (labeled **L** in Figure 3.15). There are 12 of these, 11 of which were added by hand to test the "missing building" detection capability. The remaining one, represents a significantly changed building (The cross-shaped building in Figure 3.15). All these are labeled correctly as changed or missing.

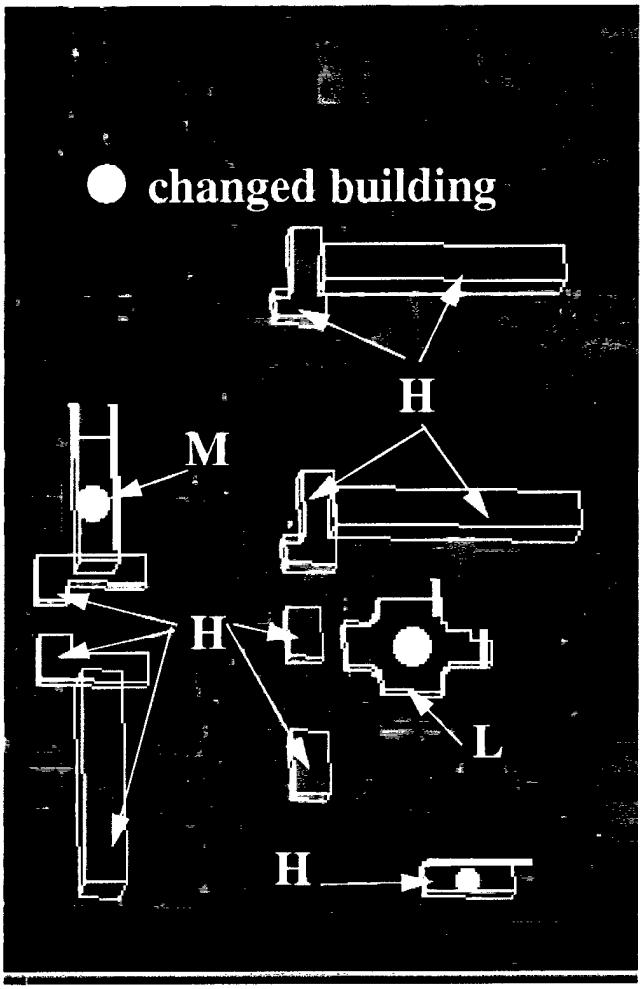


Figure 3.15 Validation and change detection result for a portion Fort Hood, Texas

3.7 Conclusion

We have shown some results and capabilities of our system for detecting and describing structural changes. It has been tested on real images (though with simulated changes to the model) and seems to be quite effective at finding significant changes in rather complex images. It is able to find missing (or misplaced) buildings, buildings with changed (or incorrectly modeled) dimensions and new buildings (or previously unmodeled buildings). This system relies on use of a single image to find changes. We anticipate that its performance would be significantly enhanced by use of multiple images as they would provide independent evidence of changes and also allow the reasoning to proceed more directly in 3-D space. Several multiple image building finders are becoming available [Noronha & Nevatia, 1996; Jaynes et al, 1994; Collins et al 1995] and could be easily incorporated in our method. Our system has been ported to an industrial laboratory for possible use in current applications.

3.8 References

[Collins et al, 1995] R. Collins, A. Hanson, E. Riseman and H. Schultz, *Automatic Extraction of Buildings from Aerial Images*, in A. Grun, O. Kubler and P. Agouris, editors, Automatic Extraction of Man-Made Objects from Aerial and Space Images, Virkhauser Verlag, Basel, pp. 169-178.

[Fua, 1996] P. Fua, *Cartographic Applications of Model-Based Optimization.* Proceedings of the DARPA Image Understanding Workshop, Palm Springs, CA, Morgan Kaufman, Publisher, Feb., pp 409-419.

[Gerson and Wood, 1994] D. Gerson and S. Wood, *The RADIUS Testbed System*, Proceedings of the DARPA Image Understanding Workshop, Monterey, CA, Morgan Kaufman, Publisher, Nov., pp 231-237.

[Huang et al., 1994] C. Huang, J. Mundy and C. Rothwell, *Model Supported Exploitation: Quick look, Detection and Counting, and Change Detection*, Proceedings of the Second IEEE Workshop on Applications of Computer Vision, Sarasota, FL, pp 144-151.

[Huertas et al., 1995] A.Huertas, M. Bejanin and R. Nevatia, *Model Registration and Validation, In Automatic Extraction of Man-Made Objects from Aerial and Space Images*, A. Gruen, O. Kuebler, and P. Agouris, Editors. Birkhauser Verlag, Switzerland, pp 33-42.

[Huttenlocher & Ullmann, 1990] D. Huttenlocher and S. Ullmann, *Recognizing Solid Objects by Alignment with an Image*, International Journal of Computer Vision, Vol. 5, No.2., Nov., pp. 195-212.

[Jaynes, et al., 1994] C. Jaynes, F. Stolle, and R. Collins, *Task Driven Perceptual Organization for Extraction of Rooftop Polygons*, Proceedings of the 1994 ARPA Image Understanding Workshop, 359-365.

[Lillestrand, 1972] R. Lillestrand, *Techniques for Change Detection*, IEEE Transactions on Computers, No.7, Jul., pp 654-659

[Lin et al., 1995] C. Lin, A. Huertas and R. Nevatia, *Detecting Buildings from Monocular Images.* In Automatic Extraction of Man-Made Objects from Aerial and Space Images, A. Gruen, O. Kuebler and P. Agouris, Editors, Birkhauser Verlag, Switzerland, pp 125-134.

[Medioni et al., 1991] G. Médioni, A. Huertas and M. Wilson, *Automatic Registration of Color Separation Films*, Machine Vision and Applications, Springer-Verlag, New York, Vol. 4, pp 33-51.

[Noronha & Nevatia, 1996] S. Noronha and R. Nevatia, *Detection and Description of Buildings from Multiple Aerial Images*, Proceedings of the 1996 DARPA Image Understanding Workshop, Palm Springs, CA, pp 469-478.

[Strat et al, 1992] T. Strat, et al., *The RADIUS Common Development Environment*, Proceedings of the DARPA Image Understanding Workshop, San Diego, CA, Morgan Kaufman, Publisher, Jan., pp 215-226

4 A System for Building Detection from Aerial Images

R. Nevatia, C. Lin and A. Huertas*

Abstract

We describe a method for detecting rectilinear buildings and constructing their 3-D shape descriptions from a single aerial image of a general viewpoint. 2-D roof hypotheses are generated from linear features by perceptual grouping. Good hypotheses are selected and then verified by computing wall and shadow evidence for them, which also provide the height information for the buildings. A 3-D reasoning process resolves conflicts among hypotheses in 3-D space. Results from several images can be integrated at a high level. An interactive system allows efficient editing of results by making use of the analysis performed by the automatic system; it also allows for some initial preparation of the data to improve results of the automatic system. Some results and their evaluation are included.

4.1 Introduction

Detection and description of buildings from aerial images remains an active area of research; an excellent collection may be found in (Grun *et al.*, 1995), some more recent work is described in (Fua, 1996; Henricsson *et al.*, 1996; Weidner, 1996). Many different kinds of inputs, such as stereo images and range images, have been used. In this paper, we focus on the use of a single image. Lack of direct 3-D information makes use of a single image more difficult, but they are attractive due to the ease with which they can be obtained. It is also our experience that many of the processes involved in single image analysis are also required for multiple image analysis (Noronha & Nevatia, 1997). Our system is restricted to rectilinear shapes with flat roof but allows for *oblique* (*i.e.* non-nadir) views. It also allows for efficient human interaction where the results of the automated system can be improved with relatively few and simple interactions before and after automated processing.

Our basic approach is to use the geometric and projective constraints to make hypotheses for the presence of building roofs from the low-level features and to verify by using available 3-D cues. As, our system is restricted to rectilinear buildings with flat roofs, they project into compositions of parallelograms. We use shadow and wall evidence to verify and reconstruct 3-D structures. The system also analyzes the 3-D structures to resolve conflicts among them. A summary of this approach and some results are given in section 4.2. In section 4.3, we describe how to integrate results from multiple images. We have also developed a methodology for efficient human interac-

tions with this system, for the purposes of editing the results or to provide some guidance prior to automatic analysis. Many errors of the automated system can be corrected (or prevented) by relatively simple user interactions. These methods and some results are described in section 4.4.

4.2 Monocular Building Detection

This system consists of several layers. At first, linear edges are detected from the image. Next parallelogram hypotheses are formed that are consistent with the projective constraints given by the viewing geometry. Promising hypotheses are selected based on some 2-D and local 3-D evidence. The selected hypotheses are verified by searching for 3-D cues using wall and shadow evidence. The verified hypotheses are examined for mutual containment and overlap and a non-conflicting set is selected which provides 3-D building models. Each model is also assigned a confidence level, computed from combinations of lower-level evidence.The early stages of this process, including hypotheses formation, selection and verification by using wall and shadow evidence have been described previously (Lin & Nevatia, 1995) The current system uses an improved hypotheses generation system and various modifications have been made to the selection and verification steps, however, the general approach remains the same and we omit further discussion of them; details may be found in (Lin, 1996)

4.2.1 Containment and Overlap Analysis

The wall and shadow verification processes examine each hypothesis individually and do not analyze the relationships among them. Thus, some verified hypotheses might overlap with or contain others. At this stage, having knowledge of 3-D allows us to check that two inconsistent structures do not occupy the same 3-D space.

When one hypothesis is contained in the other, two cases can occur, as shown in Figure 4.1 (a) and (b). In the first case, a contained hypothesis does not share any side with the containing hypothesis; here the latter is likely to be a superstructure on top of the former. We also adjust the height of the superstructure to be relative to that of the base. In the second case, the two hypotheses share some common boundaries. If the two have different heights, we consider them to be in conflict and remove the one with the lower confidence. If they have the same height, and share boundaries, the containing hypotheses is removed unless there is strong wall and shadow evidence for its *non-shared* roof boundaries.

The overlap cases also fall in two cases. If the overlapping hypotheses have the same height, it is not considered a conflict and both are retained as shown in Figure 4.2 (a). When two roof hypotheses with different building heights overlap, they conflict in 3-D space and the one with weaker evidence is removed. Note that it is possible for two building hypotheses to have overlapping footprints even if the roof hypotheses don't overlap as shown in Figure 4.2 (b).

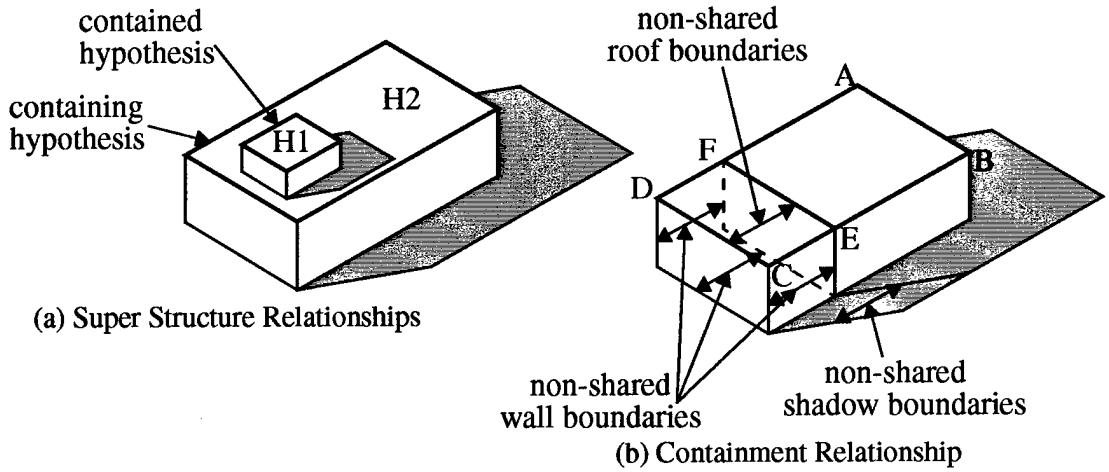


Figure 4.1 Containment analysis

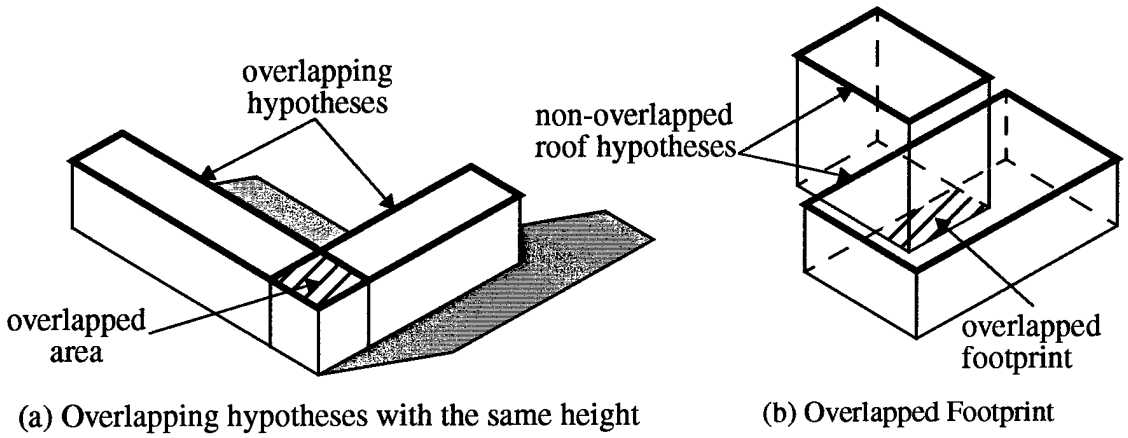


Figure 4.2 Overlap analysis

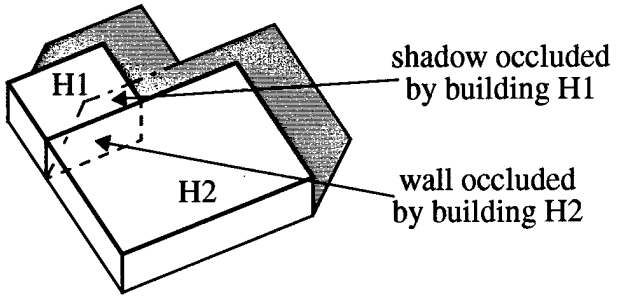


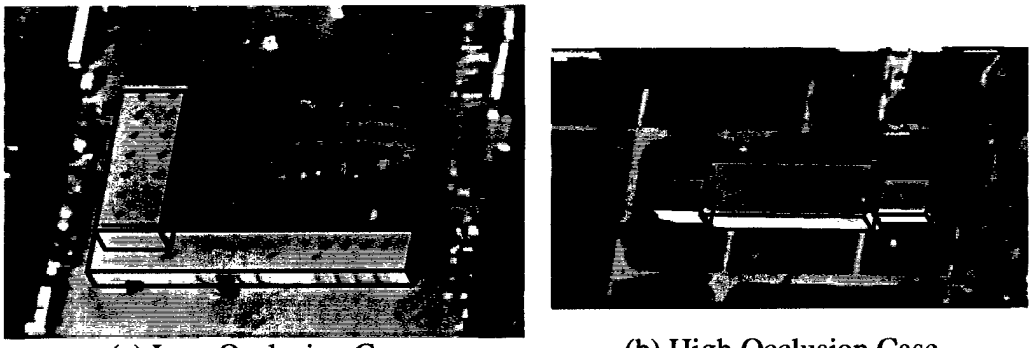
Figure 4.3 Evidence occluded by other buildings

4.2.2 Building Interaction Analysis

When nearby buildings (or their parts) occlude another, they can affect the evaluation of wall and shadow evidence of the occluded objects. In Figure 4.3 , a part of shadow evidence of hypothesis H2 is not visible because of occlusion from H1, and a part of wall evidence of H1 is blocked by H2. Say that there is enough evidence to support H2

but not H1. However, once H2 has been verified, the interaction analysis process re-evaluates H1 by examining the evidence in the non-occluded area and verifies. Verification of H1 causes confidence of H2 to be increased in turn. In general, this process may need to be iterated until no changes occur.

Figure 4.4 (a) shows the detected wire frames of the two verified hypotheses for an example. Both were detected initially, but the confidences of both parts were increased after interaction analysis. Figure 4.4 (b) shows an example where only two of the three structures of the building in the scene are detected initially; the left part is not verified as its wall and shadow boundaries are occluded by the middle part. After analyzing the occlusion, the left structure is recovered and the confidence of the middle structure is increased also, because it is occluded partly by the right structure.



(a) Low Occlusion Case (b) High Occlusion Case

Figure 4.4 Results on two examples

Figure 4.5 shows the results for a larger window (of an image from Fort Hood, Texas), containing several buildings in a complex scene viewed obliquely. As can be seen, most buildings are detected accurately. Only one has an obvious height error. No false positives are detected. Two buildings are not detected. The one in the bottom left area is not detected because of severe occlusions by nearby trees. The other is the bright building with two wings; mutual occlusions between the two parts cause both to be not verified. The two C-shaped buildings are detected but the descriptions are not accurate. The middle parts of the C-shaped buildings are not hypothesized, because there is no other evidence besides a pair of parallel lines. A part of the building in the top middle area is not detected due to occlusion and low height. There are also some structures attached to the four buildings on the left side of this image that are not detected, largely because of their low height.

It takes 877.58 seconds (14.62 minutes) to process image in Figure 4.5 on a SUN Sparcstation 20 (using the RCDE environment with all code being written in COMMONLISP). The most time consuming process, at 63% of the total, is that of parallelogram formation. The "higher-level" processes of hypothesis selection, verification and 3-D analysis take only a small fraction of the total time. The execution times are generally linearly proportional to the number of lines that are found in an image.

There are many ways to measure the quality of the results. Following (McGlone & Shufelt, 1994; Shufelt & McKeown, 1993), we use the following five measurements:

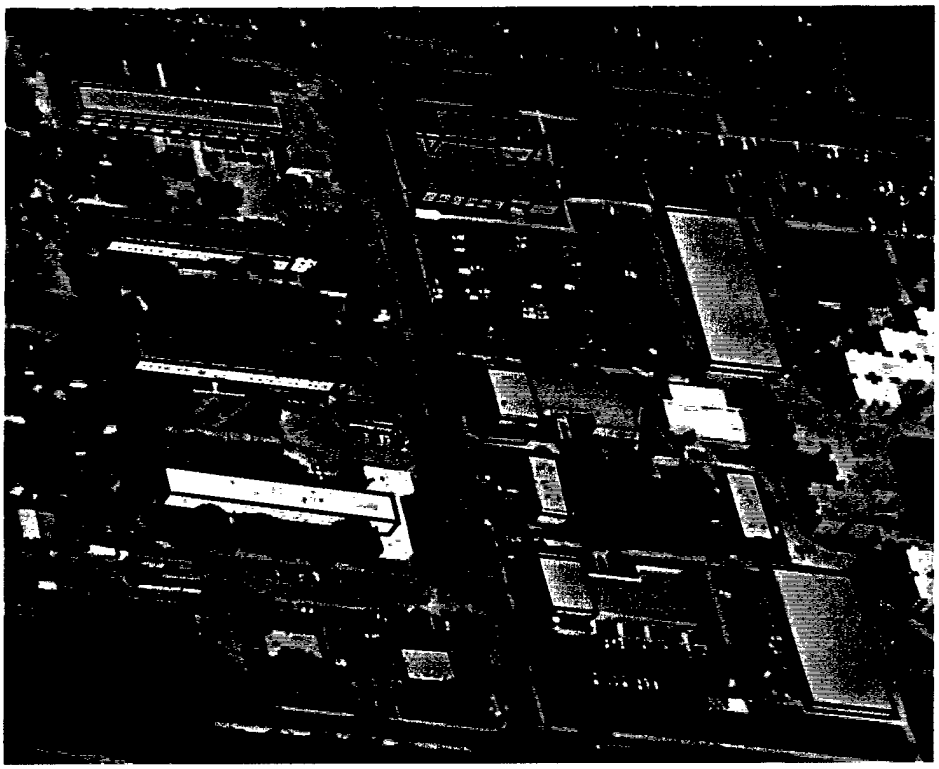


Figure 4.5 Results with multiple buildings in an oblique image of a complex scene complex scene

Detection Percentage (100*TP/(TP+TN)); Branch Factor (100*FP/(TP+FP); Correct Building Pixels Percentage; Incorrect Building Pixels Percentage and Correct Non-Building Pixels Percentage. The first two measurements are calculated by making a comparison of the manually detected buildings and the automated results, where TP (True Positive) is a building detected by both a person and the program, FP (False Positive) is a building detected by the program but not a person, and TN (True Negative) is a building detected by a person but not the program. A building is considered detected if a part of the building is detected. The accuracy of shape is determined by counting correct building and non-building pixels. These quality measurements are rather consistent for most of the images processed. Average approximate values over several examples are: Detection rate, 70%; Branch Factor,6%; Correct Buildings Pixels, 70%; Incorrect Building Pixels, 8%; and Correct Non-Building Pixels, 99%.

Another method of evaluation is to examine the number of true and false positives as a function of the hypothesis confidence; Figure 4.6 shows results for 12 windows, each containing several buildings. It should be noted that there are no false alarms for high confidence values, thus a clear choice is available between higher detection rates and lower false alarms.

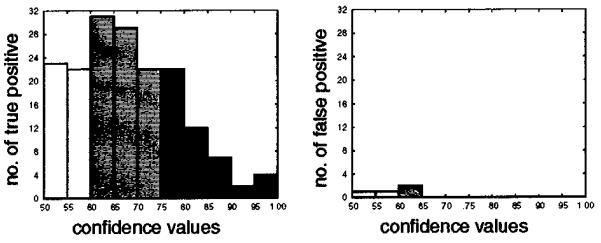


Figure 4.6 Distribution of true and false

4.3 Integration of Results from Multiple Views

Results from several views can be integrated to get more robust results. Some structures may be more reliably detected in some views depending on conditions, such as the viewing direction, the illumination direction, and the building orientation. The approach is not to perform complete stereo analysis but to merge the higher level structures only. Hypotheses in one view are projected into the other views (knowledge of relative camera geometry is assumed) and verified as any other hypotheses. If a building is correctly detected in one view, supporting evidence for it should be found in other views. On the other hand, if an incorrect hypothesis has been made, it should be unlikely to find much supporting evidence from other views. Based on this observation, a better decision can be made by integrating all evidence from all available views. A building could be verified individually in more than one view resulting in multiple hypotheses for the same structure. An overlap analysis is performed and the hypothesis with the highest combined confidence is retained. A set of 3-D models is created from the list of retained hypotheses which can be projected into any view for visualization. The situation when none of the hypotheses from any of the views is correct is not handled.

Figure 4.7 shows an example of integrating the results from two views of a building. The building is composed of three structures. The main structure in the middle is detected in the left image only and the right wing in the right image only. The left wing is detected in both images. After integration all three parts are verified and shown reprojected in the two views in Figure 4.7 . Similar improvements are obtained for other examples, such as the one shown in Figure 4.5 , but are not included for lack of space.

4.4 Interactive Editing and Preparation

While the automatic system performs well under many conditions, there are also several situations that cause it to fail to find a building or to find a correct description of the building. An interactive system has been developed to correct these errors. Many interactive systems for building detection have been developed in the past (Heller *et al.*, 1996, Neuwnschwander *et al.*, 1994). One different aspect of our approach is to use the partial results of the automatic analysis to reduce the required interactions from the user.

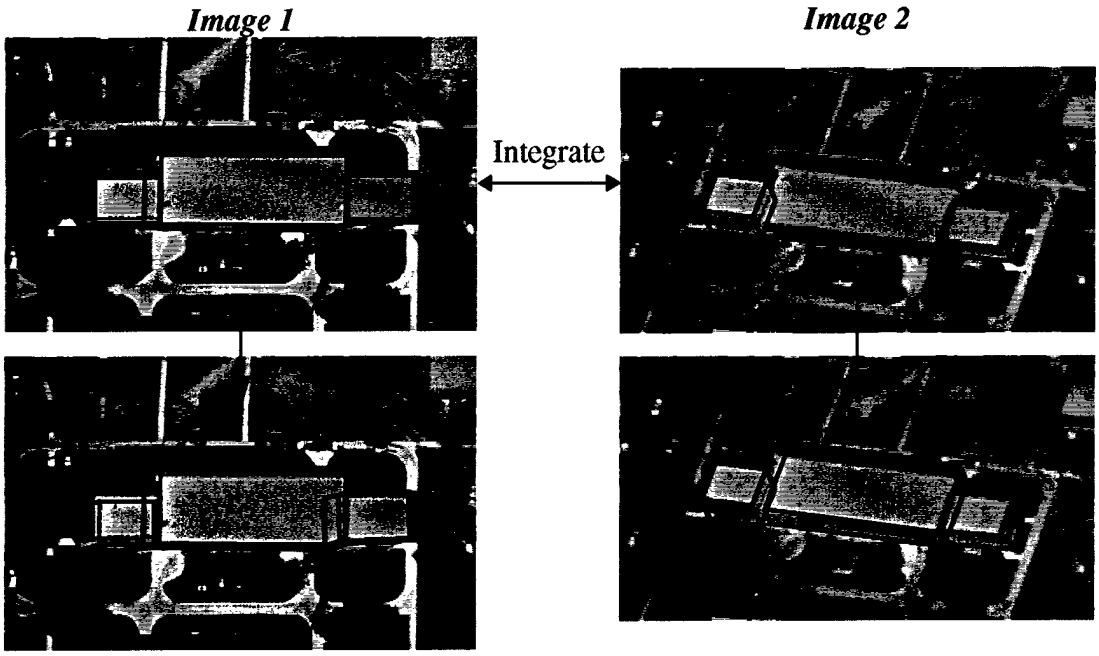


Figure 4.7 Integration of results from multiple views

There are two classes of interaction possible in this system. The first is a qualitative (or initial) interaction, the second is a quantitative (or corrective) interaction. The input for the qualitative step is simply an indication of the problem, such as a missing building and its approximate location (indicated by positioning the cursor somewhere inside the roof area). This causes the automatic system to re-examine all of the roof hypotheses generated earlier by the system and select the one with the highest score. In many cases, just this interaction results in a correct building to be detected; it was not previously output because its score was too low. A version of this system also allows to qualitatively specify the probable cause of failure (such as a dark area) which can be used in selecting the best hypotheses (see (Heuel & Nevatia, 1995) for details)

If the building detected by qualitative interaction is not correct (in dimensions, location or orientation), quantitative, corrective interactions are needed. Two ways of correcting the hypothesis are available. The user can choose to associate extracted edges and corners with a part of the building model. For example, a roof-side of the building can be specified by an edge extracted in the image. Then this edge is added to the current hypothesis (by replacing the nearest edge of the current hypothesis). Such interactions are facilitated by mouse-sensitive features of the RCDE (Strat, *et al.*, 1992). After each corrective interaction, the system forms a new parallelogram hypothesis and looks for new edges, shadow and wall evidence to support the new hypothesis. Therefore, it is possible that, after a manual correction of a roof-boundary, the wrong building height is also corrected automatically.

The user can also adjust the roof-parallelogram by dragging sides with the mouse, rotating or translating the whole model. Changes can only be made within the constraints of the building model, for example opposite sides remain parallel The extraction of a ground corner or edge (shadow corner or edge) determines the building height. These interactions are similar to a completely manual system.

We find that, in conjunction with the automatic system, relatively few and simple user interactions yield correct models. In order to complete the building detection task for the example of Figure 4.5 (14 buildings made of 29 rectangular structures), the following user interactions were required: two of the detected structures required 1 quantitative correction; fifteen qualitative interactions were required to select hypotheses for structures not detected; of these, 2, 4, 8 and 1 structures needed 0, 1, 2 and 3 quantitative interactions respectively.

Figure 4.8 shows several other building models (processed in four separate windows as shown). For this example, 38 of the structures (a rectangular building or a rectangular part) required no interactions. 27 structures were detected but required some corrective interactions (20 required one, 4 required two, and 3 required three interactions). 10 undetected structures were correctly detected with just one qualitative interaction. Remaining 29 undetected structures required 1 qualitative and 1, 2 or 3 quantitative corrections (13 required one, 11 required two, five required three). In nearly all of the cases where corrective interactions are required, only corrections of the sides and height are necessary.

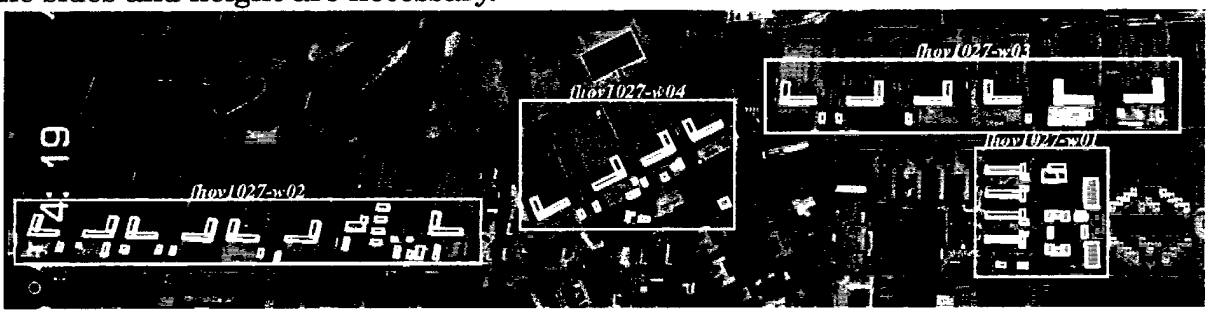


Figure 4.8 Edited results for four windows from a large image.

Initial Preparation

The performance of this system can be improved by providing the automated system with some information *prior* to its computations. In normal operation, a user would need to select images and image windows to be processed anyway. It is a relatively simple task for the user to also provide an indication of where the desired buildings are by simply pointing and clicking somewhere in the interior of such buildings and could be considered to be part of the preparation of the image data to be processed. Such input is viewed by the system in the same way as a qualitative input later, *i.e.* a building hypotheses with the highest score is always selected. Also, no buildings are output in areas not indicated by the user. This simple input greatly im-

proves the performance of the automatic system, increasing its detection rate while reducing or eliminating the false alarms.

Automatic results obtained by selecting the locations in the image of the 29 roof components is shown in Figure 4.9 (a). All roof components are detected but 14 require quantitative corrections. Eleven of these required 1 correction and the other 3 required two corrections. Figure 4.9 (b) shows the completed model. For this example, the number of structures requiring interaction is the same with initial preparation or without (as in Figure 4.5). However, the former case requires fewer corrections and takes about half as much time.

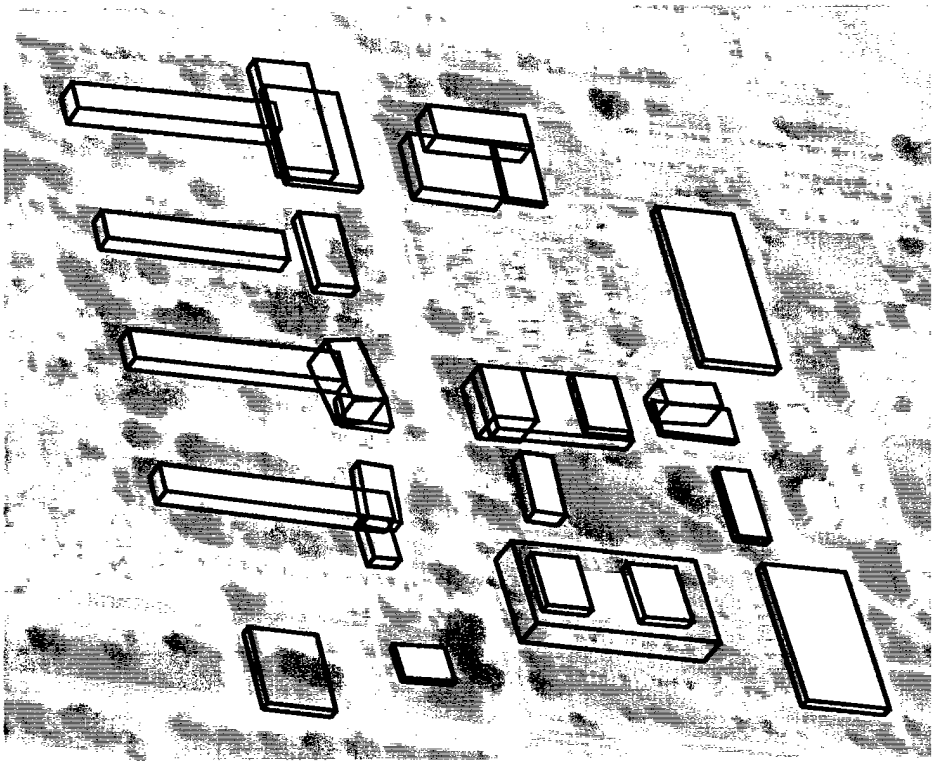
We have attempted a preliminary quantitative evaluation of this approach by comparing to the time required to construct building models in a given window by using traditional modeling tools, such as those supplied with the RCDE (Heller *et al.*, 1996). For the interactive system, we only include the time needed for initial and editing steps but not the computation time for the automated step (as it can be executed *off-line* and does not require user's attention). The results on three windows from the Ft. Hood image data set are summarized in Table 1. t_m, t_i, and t_e denote time in minutes for manual, interactive and editing processes respectively. The L and I shape data are not shown due to lack of space but are similar to those shown in Figure 4.8 . The "complex" window is the one shown in Figure 4.9 . These results compare very favorably with the manual process that would be needed for the same task. As shown in the table, the speed-ups range from a factor of about 7 to about 11, the lower number being for more complex shapes where more user interactions are required. These results are preliminary and have not been tested on large data sets with different kinds of operators (all times are for A. Huertas). Nonetheless, we believe that the indicated speed-ups are significant and offer potential for use in a practical system.

Table 2: Time Comparison (time in minutes)

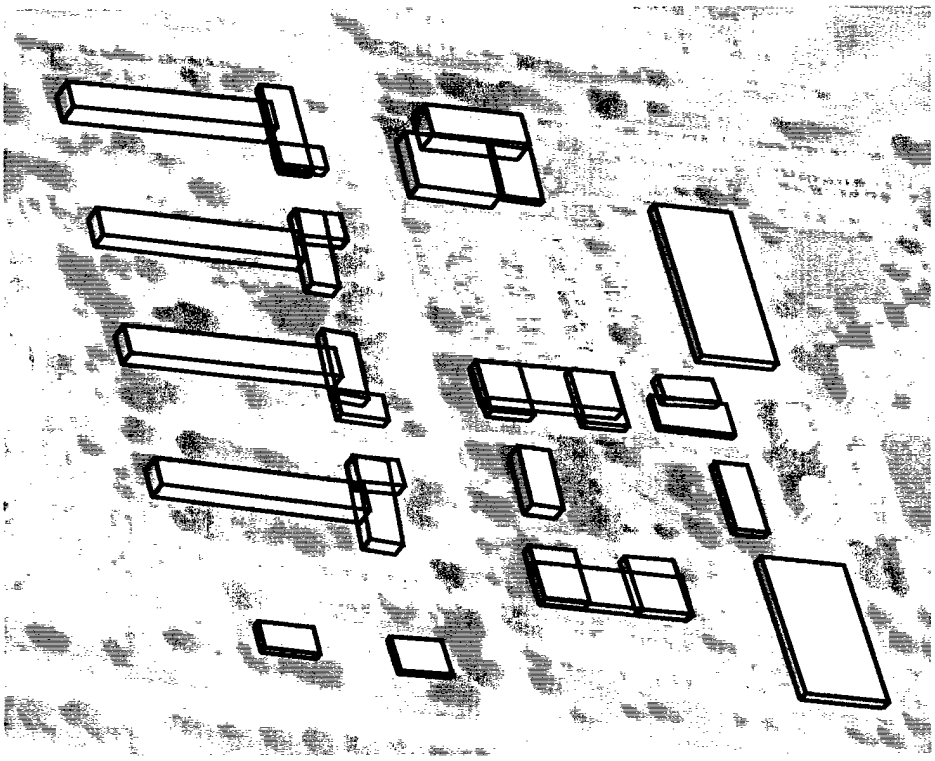
Image Description	# of Buildings	# of Boxes	t_m	t_i	t_e	# of Boxes edited	$\frac{t_m}{t_i + t_e}$
L-shape	8	12	8	0.2	0.5	2	11.4
I-shape	19	35	28	0.6	2.5	4	9.0
Complex	14	29	75	0.4	10	14	7.2

4.5 Conclusion

We have summarized our approach to automated building detection and description using a single intensity image, to integrating results of several such images, and of designing interactive tools for preparing data and editing results. The range of shapes to which these techniques can be applied remains limited but we believe that they



(a) Automated results with initial preparation



(b) Completed 3-D model

Figure 4.9 Results obtained with initial preparation and user interaction.

cover a useful and significant subset. The system has been ported to some user laboratories for further testing and evaluation.

4.6 Acknowledgments

Stephan Heuel, of the University of Bonn, developed the original version of the interactive system as a visiting researcher in our laboratory. Bill Bremner, of Lockheed Martin Corporation, suggested user provided interactions before automatic processing. Jim Pearson, of GDE Systems Inc., suggested the methodology for comparing time performance of interactive and manual systems.

4.7 References

Fua P. (1996) *Model-Based Optimization: Accurate and Consistent Site Modeling*, in Proceedings of the 18th SPRS Congress, Comm. III, WG 2, Vienna, Austria, pp. 222-233.

Grun A., O. Kubler, P. Agouris (1995) editors, *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, Virkhauser Verlag, Basel, pp. 199-210.

Henricsson O., F. Bignone, W. Willuhn, F. Ade, O. Kubler, E. Baltsavias, S. Mason, A. Grun (1996) *Project AMOBE: Strategies, Current Status and Future Work*, in Proceedings of the 18th SPRS Congress, Comm. III, WG 2, Vienna, Austria, pp. 321-330.

Heller A., P. Fua, C. Connolly, J. Sargent (1996) *The Site-Model Construction Component of the RADIUS Testbed System*, Proceedings of the DARPA Image Understanding Workshop, Palm Springs, California, pp. 345-355.

Heuel S., R. Nevatia (1995) *Including Interaction in an Automated Modeling System*, Proceedings of the IEEE Symposium on Computer VIsion, Coral Gables, Florida, pp. 383-388.

Lin C., A. Huertas and R. Nevatia (1995) *Detection of Buildings from Monocular Images*, in A. Grun, O. Kubler, P. Agouris, editors, Automatic Extraction of Man-Made Objects from Aerial and Space Images, Virkhauser Verlag, Basel, pp. 125-134.

Lin C. (1996) *Perception of 3-D Objects from an Intensity Image using Simple Geometric Models* Ph.D. Dissertation, Computer Science Department, University of Southern California.

McGlone J., and J. Shufelt (1994) *Projective and Object Space Geometry for Monocular Building Extraction IEEE Proceedings of Computer Vision and Pattern Recognition*, 54-61.

Noronha S., R. Nevatia (1997) *Building Detection and Description from Multiple Aerial Images* to appear in *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, San Juan, Puerto Rico.

Shufelt J., D. McKeown (1993) *Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery Computer Vision, Graphics and Image Processing*, 57(3): 307-330.

Strat T., L. Quam, J. Mundy, R. Welty, W. Bremner, M. Horwedel, D. Hackett, A. Hoogs (1992) *The RADIUS Common Development Environment Proceedings of the 1992 DARPA Image Understanding Workshop*, San Diego, California, 215-226.

Weidner U. (1996) *An Approach to Building Extraction from Digital Surface Models* Proceedings of the 18th SPRS Congress, Comm. III, WG 2, Vienna, Austria, pp. 924-929.

5 List of Publications

G. Guy and G. Medioni, Inferring Global Perceptual Contours from Local Features, *International Journal of Computer Vision. Vol. 20*, no. 1/2, pp. 113-133, 1996

S. Han and G. Medioni, "3D Digitizing Made Easier by Unstructured Sketching", SIGGraph 97, Los Angeles.

S. Han and G. Medioni, "3DSketch: Modeling by Digitizing with a Smart 3D Pen", ACM Multimedia'97.

S. Han, M. Lee, and G. Medioni, "Non-uniform Skew Estimation by Tensor Voting", IEEE Document Image Analysis, Puerto Rico, June 20, 97.

S. Han and G. Medioni, "Edge-aligning Surface Fitting Using Triangular B-Splines", DARPA Image Understanding Workshop, New Orleans, May 1997.

S. Han and G. Medioni, "Triangular NURBS Surface Modeling of Scattered Data," *IEEE Visualization*, San Francisco, California, USA, Octocber 1996.

S. Han and G. Medioni, "Spherical Winged B-Snakes", IEEE intl conf. on Image Processing, Lausanne, Switzerland, Sep. 1996

P. Havaldar, G. Medioni, and F. Stein, "Perceptual Grouping for Generic Recognition", *International Journal of Computer Vision. Vol. 20*, No. 1/2, 1996, pp. 59-80.

A. Huertas and R. Nevatia, "Detecting Changes in Aerial Views of Man-Made Structures," to appear, IEEE Intl. Conference on Computer Vision, Bombay, India, January 1998.

C. Lin and R. Nevatia, "Building detection and description from intensity images", accepted for publication in Computer Vision and Image Understanding, 1998.

S. Noronha, R. Nevatia, Detection and Description of Buildings from Multiple Aerial Images, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, San Juan, Puerto Rico, pp. 588-594. and in DARPA Image Understanding Workshop, New Orleans, LA, pp. 989-998.

M. Zerroug and R. Nevatia, "Part-based 3-D descriptions of complex objects from a single image", submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, August 1997.

M. Zerroug and R. Nevatia, "Volumetric descriptions from a single intensity image," *International Journal of Computer Vision. Vol. 20*, No. 1/2, 1996, pp. 11-42.

6 Professional Personnel

6.1 Personnel

Professional personnel included Dr. R. Nevatia, Dr. G. Medioni, Dr. K. Price, and A. Huertas.

6.2 Ph. D. Graduates

In this contract period we have had 2 Ph. D. graduates: S. Han and C. Lin.